DYNAMIC COMPOSITION OF SERVICE SPECIFIC OVERLAY NETWORKS

Yousif Al Ridhawi

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the PhD degree in Electrical and Computer Engineering

School of Electrical Engineering and Computer Science (EECS)
Faculty of Engineering
University of Ottawa

Abstract

Content delivery through service overlay networks has gained popularity due to the overlays' abilities to provide effective and reliable services. Inefficiencies of one-to-one matching of user requirements to a single service have given rise to service composition. Customized media delivery can be achieved through dynamic compositions of Service Specific Overlay Networks (SSONs). However, the presence of SSONs in dynamic environments raises the possibility of unexpected failures and quality degradations. Thus constructing, managing, and repairing corrupted service paths are challenging dilemmas. This thesis investigates the problem of autonomous SSON construction and management and identifies the drawbacks of current approaches.

A novel multi-layered, autonomous, self-adaptive framework for constructing SSONs is presented. The framework includes a Hybrid Service Overlay Network layer (H-SON). The H-SON is a dynamic hybrid overlay dedicated to service composition for multimedia delivery in dynamic networks. Node placement in the overlay depends on the node's stability, types and quality of provided services. Changes in stability and QoS of service nodes are reflected by dynamic re-organizations of the overlay. The H-SON permits fast and efficient searches for component services that meet client functional and quality expectations.

Self-managed overlay nodes coordinate their behaviors to formulate a service composition path that meets the client's requirements. Two approaches are presented in this work. The first illustrates how SSONs are established through dynamically adaptable MS-designed plans. The imprecise nature of nonfunctional service characteristics, such as QoS, is modeled using a fuzzy logic system. Moreover, semantic similarity evaluations enable us to include, in compositions, those services whose operations match, semantically, the requirements of the composition plan.

Plan-based composition solutions restrict service discovery to defined abstract models. Our second composition approach introduces a semantic similarity and nearness SSON composition method. The objective is to free service nodes from the adherence to restrictive composition plans. The presented work illustrates a service composition solution that semantically advances service composition paths towards meeting users' needs with each service hop while simultaneously guaranteeing user-acceptable QoS levels.

Simulation results showcase the effectiveness of the presented work. Gathered results validate the success of our service composition methods while meeting user requirements.

Dedication

There are a number of people without whom this thesis might not have been written, and to whom I am greatly indebted and dedicate this dissertation.

To both of my parents, albeit I fully understand any amount of gratitude shown to them is woefully inadequate.

To my father, who taught me that knowledge is the greatest treasure one can possess. His sage advice and constant encouragement have given me the courage to work and to hold tight to my position no matter how hard life gets.

To my mother, from whom I learnt the power of eternal love; she has always been a source of inspiration throughout my life. Her selfless sacrifices for my brothers and I shall never go unacknowledged. From her I learned that even at the darkest moments in life, the light of hope can shimmer from one's heart to illuminate a whole new world.

To my loving wife, whose incessant encouragement, patience, and love made it all possible. Thank you for accompanying me during my ups and downs. You have always been my pillar, my joy, and my guiding light without which this task would have been impossible to complete.

Acknowledgements

I am deeply indebted to my supervisor, Dr. Ahmed Karmouch. He has not only influenced my graduate studies, but my whole life. His deep commitment to research and teaching have instilled in me a strong sense of discipline and integrity, for which I am eternally grateful.

I would like to thank *Cistech Limited* for their support and the constructive feedback that helped improve the quality of this thesis.

I would also like to thank the many members of the *Intelligence for Mobile Autonomic and Cognitive Networks Laboratory* for their help and support during my research period. Their companionship and friendship eased many tasks during times of difficulty and have brought joy to my life.

My thanks extend to Ibrahim Al-Oqily whose research material was the stepping stone for my thesis work.

Last but not least, I would like to thank my brother, Ismaeel, for his constant inquisitions about the details of my work. His curiosity has helped me shed light onto many unanswered questions and forced me to strive to find new solutions.

Contents

List of Figures

Acronyms

ACL	Autonomic Control Loop
ACon	Autonomic Controller
ACS	Abstract Control Space
AM	Autonomic Manager
AN	Ambient Network
ARI	Ambient Resource Interface
ASI	Ambient Service Interface
ASI	Autonomic System
BPEL	Business Process Execution Language
CA	Component Agent
CM	Context Manager
CoRE	Component Runtime Environment
CRP	Churn-Resilient Protocol
CSA	Composition Agent
DHT	Distributed Hash Table
E2E	End to End
EM	Energy Manager
EP	Energy Profiler
FA	Functional Area
FCFS	First Come First Serve
FCL	Fuzzy Control Language
FIS	Fuzzy Inference System
FL	Fuzzy Logic
GSO	Global Skeletal Ontology
H-SON	Hybrid Service Overlay Network
KP	Knowledge Plane

LB	Limited Broadcast
LM	Link Major Node
MC	Media Client
MH	Message Handler
MP	Media Port
MPS	Media Port Sensor
MS	Media Server
MUI	Multimedia User Interface
NGSON	Next-Generation Service Overlay Network
NM	Normal Major Node
OB	Overlay Builder
OCS	Overlay Control Space
ONode	Overlay Node
OSL	Overlay Support Layer
OWL	Web Ontology Language
P2P	Peer to Peer
PM	Performance Manager
QH	Quality of Service Handler
QoE	Quality of Experience
QoS	Quality of Service
RM	Resource Manager
RON	Resilient Overlay Network
SeGSeC	Semantic Graph-Based Service Composition
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SOC	Service Overlay Computing
SON	Service Overlay Network
SSON	Service Specific Overlay Network
UFM	User Feedback Manager

WS-BPEL	Web Service Business Process Execution Language
WSCI	Web Service Choreography
WSDL	Web Service Description Language

Chapter 1

Introduction

1.1. Overview

Today's service-centric paradigm of networking and QoS-based content delivery has resulted in a plethora of new services. To a large extent, most software capabilities have become deliverable and consumable services. The Internet has surpassed the point of providing host connectivity and has become a global platform for sharing service components and content. The level of functionality abstraction has progressively increased from modules, to objects, to components, and now to services [1]. As such, Service-Oriented Computing (SOC) has become a standard for service users to compose services [2] [3]. SOC focuses on service composition for application development that is based on service publication, registry, and discovery by service requesters. The formal definition of Service Oriented Architecture (SOA) states that a SOA is "*a set of components which can be invoked and whose interface descriptions can be published and discovered*" [4]. Services in SOA are viewed as building blocks that combine information and behavior, hide internal workings, and present a simple interface for the rest of the system. The SOA must ensure that services do not get reduced to simple interfaces. Rather, they should have a personal identity, and can be managed individually and in sets. These sets are referred to as compositions.

With today's wireless technology advancements and rapid deployment of mobile devices, the SOA paradigm has begun its shift from wired to wireless networks. However, wired-based service composition architectures were not designed with mobility taken into

consideration. The seamless composition of distributed service components into more complex applications in a dynamic environment is an especially laborious process. This is due to the possibility of facing disruptions caused by movement of users and service providers. Additionally, the heterogeneity of devices, resource variability and service reliability is highly unpredictable in mobile networks. Furthermore, the presence of multiple service providers offering the same service results in more conglomerate composition processes.

With increasing mobility and numbers of available services, the problem of linking and composing desolated individual services into a distinguished complex service becomes progressively complex. Overlays initiated the distribution of stream applications such as multicasting, content distribution networks, and peer-to-peer file sharing [5] [6] [7] [8] [9]. This was effectively performed by using the Internet as the lower-level infrastructure to provide higher-level services. As such, it became possible to provide, through overlays, new services to end users. OverQoS [10] and Resilient Overlay Networks (RON) [11] were two early examples of utilizing overlay networks for managing and allocating overlay links and resources to enhance service performances. The former proposed an overlay-based QoS architecture to enhance the Internet's general QoS. The latter was designed to increase the Internet's speed in detecting and recovering from path outages or degraded performance. However, today's use of overlays has expanded vertically in the OSI architecture towards the application layer. The new goal is to focus on easily compose-able media services and network-side services to help streaming applications achieve QoS guarantees. Nonetheless, service compositions in an unstable dynamic environment is a challenging problem keeping in mind the goal of maintaining the required QoS levels when forming composition paths.

In this dissertation, we address the problem of building and managing autonomic service specific overlay networks for media delivery in dynamic networks. This chapter briefly discusses the different aspects of the autonomous formation of a service composition path between a media source and a media client in a dynamic environment. The system design process and architecture is also described. Furthermore, we outline the contribution of our work in comparison to current research in literature. Ultimately we present the organization of the remainder of the thesis.

1.2. Service Composition over Service Overlay Networks (SONs)

With the invasion of countless personal devices with different sets of features, and with the massive quantities of multimedia content available, seamless delivery becomes a challenge. For media content to be specifically tailored to user QoS requirements and device capabilities, network-side services can be used. Given the probable lack of a single service to perform all required media adaptations, the concept of service composition has been introduced. This refers to a mechanism that allows individual services to collaborate in order to create new services [12]. Using this concept, users can request a multimedia stream and the network can transparently link the service functions needed. This can be done by constructing overlay networks designed specifically to meet user requirements.

Delivering customized media by providing advanced services through service composition and forming Service Specific Overlay Networks (SSONs) requires dynamic construction of overlays without prior knowledge of the underlying physical network. These media services include dropping frames to meet QoS constraints, caching media content before being viewed and converting media formats to meet the clients' needs. The idea behind SSONs emerged from SONs [13]; application-layer logical networks formed by links between service nodes with matching inputs and outputs. SONs were an effective method to address some of the end-to-end QoS issues that plagued the Internet. Each SON node provided application-level data routing as well as value-added services such as media transcoding and data encryption. Each service was offered via Service Level Agreements (SLAs). The architecture was based on services gateways connected by the underlying network domain with bandwidth guarantees.

However, most proposed work in media routing and delivery of the past had focused on solving one specific problem at a time. These work focused separately on media adaptation, caching architectures, or multicast protocols. Additionally, another problem emerged in determining where data transformation should take place? Placing the burden on the end-user devices would be unreasonable. This is especially the case since we are assuming these devices to be mobile, hence the limitations in available resources such as processing power,

battery life, and performance level. Similarly, requiring the service provider to handle all required media transformation is unrealistic. Considering the broad range and heterogeneity of end-user devices, it would be impossible for every service provider to handle all the necessary media transformations. Consequently, the remaining option is to distribute these media transformations over a number of network-side service nodes.

Enter SMART (Multimedia Routing and Transport based on Service-Specific Overlay Networks) [14], which enhanced media services by taking advantage of network-side media processes called MediaPorts (MPs). These MPs could perform caching, adaptation, synchronization and media routing services. The development of SMART multimedia delivery framework enabled the transparent inclusion of MPs in end-to-end paths while taking advantage of value-added functions inherent to autonomous networks. SMART allowed adaptations of media delivery services based on network, user and service context information and policies. With SMART's SSON guideline, a flexible and transparent construction of end-to-end media delivery paths can be made from MediaServers (MS) to MediaClients (MC). An SSON is constructed independently for each media delivery session. Consequently SSONs may negatively affect each other, thus resulting in bottlenecks and performance degradations. To solve the problem, an overlay network management mechanism was introduced in [15] to reduce the complexity of managing overlays and preserve their operations.

1.3. SSON Composition Problem

The progression of service composition from wired to wireless networks has amplified the possible benefits gained in SSON creation for new complex services to users on the move. However, this has come at the cost of increased complexity and overhead. Mobile networks are characterized by unpredictable movement of nodes and limitations in resources. Moreover, the costs associated with forming SSONs and managing them increase as SSONs may overlap and negatively affect each other. In any service composition process, there are a set of prerequisites that must be acknowledged. First it must be determined what services are needed to perform the required series of media modifications. Thus, there must be a

clear progression map clarifying the path that should be followed from the MS towards the MC. Secondly, forming the above composition plan requires an understanding, even if unspecific and broad, of the services available in the network. Proposing an SSON path that requires non-existent component services may result in increased growth of message overhead and wasted bandwidth without the ability to find the desired MPs.

SSON construction is further complicated by a third problem: which services can link and interoperate with each other? It is essential to incorporate a mechanism that can model and interpret inter-service relationships to successfully form an SSON. Linking consecutive MPs in a composition involves determining syntactic and semantic details of the output of one MP relative to the input of the next MP. Finding a service that performs a required function does not guarantee it can seamlessly link with the previous or next MPs without having their respective ports associability challenged. In completely distributed SSON compositions, reliance on composition plans becomes impractical. Therefore, a plan-free service composition approach must be developed. It is essential that semantic knowledge should be incorporated into the composition process to find a plan-free methodology to compose services automatically without human intervention.

Moreover, the SSON's ability to recover from MP failures is another obstacle that must be overcome. The dynamic changes in network conditions due to node failures and link congestions may quickly render the current established paths and management information obsolete. Changing loads on MPs and resource depletions may result in unexpected degradations in service performance. Thus a safe-failure and recovery processes to reconfigure the affected SSONs must be effectively established. The above complications must be handled quickly and seamlessly without impacting the MC's perceived media flow.

It must be acknowledged that these requirements must be handled alongside maintaining an acceptable level of QoS, bandwidth, and latency. The specifications of these parameters are determined by the MC's direct request, the MS's abilities, or the network's current status. Therefore, any new service composition approach must cover all phases needed for building an SSON. It must formulate a composition plan, search for potential MPs, build an SSON, and handle all end-to-end management paths established from MSs to MCs.

Furthermore, the SSON composition problem must take into consideration two

contradicting perspectives; the users' and the service providers'. Users are mainly concerned with acquiring their media and services at the best quality level they can handle and at the lowest possible price. Contrarily, service providers are concerned with providing users' with the desired services while efficiently utilizing local resources. Thereupon, the utmost goal of a service composition process is to find service providers that can guarantee the highest level of QoS, charge the lowest price, have the shortest delay, and achieve the highest level of user satisfaction while maximizing their own profits.

1.4. Motivation

Service composition has received much attention over the past decade. Much progress has been made since the expansion of the field of distributed web services and their goal of providing platforms and languages that can allow the easy integration of heterogeneous systems. Despite all efforts, the task of service composition has only increased in complexity. The scope of services has expanded beyond that of mere web services to include network-side functions such as synchronization, storage, media modification and routing. The number of available services has increased dramatically during recent years and the numbers are only expected to go higher. Additionally, the fast pace of service creation, which can occur on the fly, has made the assumption regarding the presence of searchable and updatable central repositories inconceivable and un-scalable. Furthermore, the unguided process of service development by different organizations using inconsistent modeling approaches has bestowed us with a lack of a universal language to define and evaluate services in a unified manner.

To add more complexity to the problem, service composition research has now shifted towards mobile environments. Service providers' movements are unpredictable, and QoS levels and bandwidth availability are constantly fluctuating. Moreover, users' devices are becoming more powerful and their expectations of quality level have surged even further. Therefore we are motivated to search for a new solution that can find a suitable mechanism to discover mobile media processing functions and seamlessly integrate them into media delivery sessions. Furthermore, the established sessions must be monitored and adapted

dynamically to network, user, and service providers' changing conditions.

The inflexibility of the routing infrastructure of the Internet and its resistance to fundamental changes has led to the use of overlay networks as a means to provide additional flexibility and control. SMART assumes the presence of a SON over which SSONs are constructed. However, SMART does not specify the means of constructing and managing the SON. Nevertheless, topologies are the most prominently configurable components of overlay networks. Topologies can be dynamically configured to accommodate contextual and communication requirements that vary over time. Overlay topologies can be divided into two distinct classes; structured and unstructured. The former provide efficient support for simple exact-match queries but constrain their overlay topology to achieve this. Unstructured overlays do not constrain overlay topology or query complexity since they rely on flooding or random walks to discover data. Therefore it is essential to provide an SON topology that has low maintenance costs, permits exploiting the heterogeneity perceived in mobile devices over which the overlay is constructed, and supports complex queries to efficiently compose SSONs.

A great challenge is the case of composite services described by abstract processes where the aim is to find the best combination for concrete services to replace these processes. This becomes even more challenging if QoS must be considered at run time heeding the dynamicity of mobile environments. A major limitation of most existing service composition solutions arises from their adoption of a centralized approach to service composition, and lack of an efficient and methodical QoS management.

The use of fuzzy logic has long been used to manage and model imprecise concepts where definitive values cannot be found. As such, the use of fuzzy logic has spread to modeling unpredictable variations in QoS [16] [17] [18] [19]. This is particularly the case in dynamic networks where node stability and network traffic are highly unpredictable. As such it would be reasonable to encompass a fuzzy-based reasoning system when constructing SSONs that comply with the MCs' QoS requirements in unstable dynamic environments. The fuzzy system can dynamically adjust the QoS acceptability scale as an SSON is formed and while it is being managed during its activity.

Service composition requires interoperability between diverse component services. Many existing web service languages defined standards for service discovery, description, and messaging. However, several problems have not been fully covered especially for compositions in distributed environments. The dynamic composition of services requires the recognition of services that can be affiliated with each other to create a composition. This process requires finding an automated method that bypasses human intervention such that an interpretation of the semantic and syntactic descriptions of services is automatically comprehended. Many markup languages have been suggested for service descriptions [20] [21] [22]. However, the means of utilizing these languages to evaluate inter-service relationships and similarity have barely been researched to permit autonomic compositions.

Finally, the instability of mobile environments compels us to find an autonomic re-configurability mechanism for SSONs. Changes in the initial user requirements, unpredicted node failures, and changes in network load and loads on particular service providers, all must be taken into consideration. It is disadvantageous to limit SSONs management to the creation phase and not acknowledge the need for management during the SSON's validity during its lifetime. Therefore, there is a need to find a precise mechanism to manage established SSONs and dynamically reconfigure them when needed.

The aforementioned limitations of current service composition mechanisms exemplify a set of strong motivations for developing a novel autonomous SSON composition mechanism with dynamic capabilities. The composition mechanism must formulate composition schemes from the users' requirements, discover potential service providers, and form respective composition paths. The mechanism must autonomically discover, compare, and evaluate the syntactic and semantic characteristics of services. Moreover, the mechanism must perform the above responsibilities in a continuously changing environment without any human intervention.

1.5. Dissertation Overview

This thesis approaches the issue of autonomic SSON composition from two levels; the SON level, and the SSON level. The first level is concerned with building a dynamic and

efficient overlay structure that easily permits service compositions. The second level is concerned with finding a dynamic mechanism for creating user-specific SSONs keeping in mind the dynamicity of user, network, and service provider's requirements and contextual information.

The SON formation phase has the task of efficiently forming an overlay topology. The overlay must consider and adapt to the dynamicity and instability of mobile networks. Furthermore, the overlay should avoid scalability and single point of failure issues of centralized solutions, while also avoiding the large overhead associated with queries in unstructured overlays. Moreover, the overlay should adopt a structure that aids in rapid, accurate and efficient searches for services during the composition phase.

The SSON composition phase must be designed as a dynamic process where component services are composed at runtime rather than statically at system initialization. Users' functional and non-functional requirements must be considered and obeyed to increase user satisfaction rate. Composition must have low delay, maintain a smooth multimedia delivery, and handle failures swiftly. Any failure should result in a path reconfiguration process that minimizes adverse effects on user experience.

Incorporating these aspects is achieved through a hybrid SON structure that organizes service node inter service-related clusters. Furthermore, the instability of mobile nodes, and the variations in the quality of their provided services and available resources are reflected directly in radial distance of these nodes in the overlay as well as the role associated with them. In concurrence, the SSON composition process is enhanced with fuzzy logic inference capacity. This permits the system to evaluate the required and available QoS properties and analytically ascertain its decisions regarding the validity of a service in a composition path. Additionally, an ontology-based semantic similarity and nearness decision engine can evaluate the similarity between services to link a logical composition path.

SSON construction is a three-part process that begins with services discovery and composition, followed by monitoring and adaptation. The focus of this dissertation is the first stage of discovery and composition. The goal is to create an SSON characterized by stability and QoS levels that meet MC expectations. In this work we present a unique MP

discovery and SSON composition method. Our solution meets the challenges described above and guarantees a decentralized, stable, best-fit and reconfigurable service composition for MC requests in unstable dynamic networks.

We clearly focus in this dissertation on the process of service discovery and composition that can meet incoming MC requests. The dynamicity of mobile networks implies that MPs' movements can adversely affect the current QoS of established SSON. Furthermore, unexpected departures of MPs can break existing SSON paths. Maintaining an SSON path's QoS during its lifetime is not the focus of our work. Such process requires monitoring and dynamically adjusting SSONs after their composition as network conditions fluctuate [23] [24] [25]. Monitoring QoS of transmitted media, retransmission of lost data, and the recovery of failed links are the responsibility of the application layer. We are in this dissertation concerned with the discovery of MP services and construction of most stable SSONs given the MC's QoS requirements and expected media delivery duration.

The objectives of the presented research work can be summarized as follows:

QoS-Based Composition: composing an SSON should be based on QoS criteria that meet the users' expectations thus increasing the system's satisfaction rate.

Increased autonomy: human intervention should be kept at a minimum. This increases the system's speed, shields system administrators from unnecessary details, and increases the system's extensibility.

Scalability: The performance of the hybrid SON layer should be maintained regardless of the number of nodes joining and leaving the overlay. Furthermore, the performance of the SSON composition process should be controlled even when the number of requests increases from incoming users.

Handling instability: the system must handle established SSON paths even with unexpected arrivals and failures of nodes. Service should be provided seamlessly to users, and it should be possible to dynamically recover disconnected SSON paths.

Flexibility: the composition process should handle the imprecise nature and dynamic changes occurring in QoS specifications.

1.6. Summary of Contributions

The goal of this dissertation is to investigate new principles and design new models for SSON autonomous composition and management. The major contributions of this dissertation can be summarized as follows:

1. **A Dynamic and Self-Managed Service Overlay Network.**
   Design and specification of a hybrid SON layer. The overlay's topology aids in simplifying the SSON composition process in the above layers. The hybrid SON respects and utilizes the autonomic behavior of its member nodes and is resilient to node failures. The overlay makes use of the types of service provided by joining MPs, dynamically monitors changes in QoS levels, and automatically adjusts the semantics and QoS levels of MPs are used to achieve the highest possible performance.

2. **A Plan-Based Service Discovery and SSON Composition Scheme.**
   Design of a novel service composition process for SSON creation. The composition scheme is built over our hybrid service overlay network and uses a unique fuzzy-based system to evaluate the validity of non-functional QoS requirements. Moreover, the composition method utilizes an ontology-based semantic similarity measurement procedure that increases the accuracy of selected MPs thus meeting user expectations.

3. **A Unique MP Search Algorithm for QoS Enhanced Service Discovery.**
   MP service queries utilize our unique hybrid SON layer, and our fuzzy-enhanced and semantic nearness-based SSON composition process. The search method is characterized with reduced message overhead, reduced search time, and higher level of accuracy than existing approaches. Furthermore, the search method does not rely on a centralized service registry and can recover from node failures quickly and with the highest performance.

4. **A Plan-Free and Decentralized Service Specific Overlay Network Composition.**
   Design of a novel plan-free service composition process. The scheme is characterized by its flexibility and ability to be implemented over any service

discovery mechanism. We introduce a semantic similarity and semantic nearness evaluation methodologies. The process guarantees any SSON composition will semantically progress towards the MC's requirements with every hop. The system does not require service registration, composition plan formation, or service-based MP distribution in the network. Simulation tests illustrate the validity and usefulness of the plan-free mechanism.

1.7. Organization of the Dissertation

The remainder of the dissertation is organized into the following chapters.

Chapter 2 presents some of the related work that forms the background history of our work.

Chapter 3 presents state of the art research currently adopted, we identify some of the issues addressed by various research group and the limitations of their work in service composition.

Chapter 4 describes an autonomous SSON construction multi-layered architecture. The chapter outlines the design, goals, and responsibilities of each layer and their interaction process. A hybrid SON network is presented some simulation results are also presented to demonstrate the performance of the hybrid overlay.

Chapter 5 presents our novel service composition process. A plan-based solution is presented, illustrating how a service request initiated by the MediaClient can be served by the MediaServer. The request is divided into a set of partial services with specific QoS levels derived from the MediaClient's request is formed. The plan is utilized to search for, choose, and link candidate services to form the completed composition path.

Chapter 6 describes our fuzzy-logic enhanced MediaPort search mechanism. The uncertainties associated with QoS properties are reflected in our fuzzy evaluation system. The results are utilized to determine the sector bands of our H-SON where composition queries should be directed.

The process limits the number of messages sent over the network and reduces the SSON composition delay. Simulation results are provided for the proposed solution.

Chapter 7 describes our plan-free SSON composition mechanism. An autonomic semantic- and syntactic nearness-based candidate MP selection algorithm for service compositions is presented. The algorithm is completely independent of the service discovery mechanism and focuses on the process of MP selection. The presented work can be applied to most available resource and service discovery mechanisms to enhance MP selection with increased accuracy and minimum delay. Simulation results provide a clear proof of the feasibility of the presented SSON composition method with existing service discovery mechanisms.

Finally, Chapter 8 summarizes the research work and presented contributions, and discusses directions of the future research plan.

Chapter 2

Background

Service composition is a process that requires the orchestration of a number of existing services to provide a composite service that is richer and assembled in a manner that meets user requirements. A considerable amount of research has been conducted in the field of service composition and other intertwined topics. Service composition techniques are usually designed as an extension to service discovery. Moreover, service composition is focused on dynamically forming overlay networks using network-side service nodes.

This chapter presents some of the background information onto which our work builds and extends to provide an efficient solution to discovering network-side services, composing them into complex service overlays to meet MCs' requirements. As such, one possible solution to service composition can be divided into two levels:

1. Designing an overlay network encompassing all service nodes present in the environment. The overlay should allow for fast and efficient service discovery, have the dynamicity required in mobile networks, and permit service composition according to specified quality levels.
2. Find an efficient, accurate, QoS-enabled, and dynamic method for discovering service nodes in the overlay network of step 1. The composed service must meet the client's requirements.

In this assertion we introduce two service composition solutions. The first is based on the two steps described above and relies on a preconceived composition plan shared by MPs during the composition process. The second is devoid of any composition plans. The

solution relies on a complex process of semantic and syntactic comparison evaluations. Details pertaining to the two composition approaches are to be discussed in the subsequent chapters.

This chapter presents an overview of existing and ongoing research in the fields related to service overlay design (step 1 above) onto which service-specific overlays are composed. We focus on a number of topics including the SMART architecture and the SORD service discovery system, as well as some of the existing overlay network designs. We discuss some of the limitations present within existing work, and the reasons it has lead us to present a new service overlay network onto which service paths are composed.

2.1. Smart Media Routing and Transport (SMART)

The Ambient Networks Integrated Project [26] attempted to face the challenge of forming a common control layer for joining services of multiple networks. Integration should cover multiple devices, multiple networks, and multiple access technologies. Ambient Networks (AN) aimed to provide a domain-structured, edge-to-edge view for the network control. Consequently, a homogeneous view of the network appears to users of network services. Therefore users can own/operate special, low-complexity networks.

ANs are derived from all-IP mobile networks with a clear separation between transport and control related tasks. These tasks are grouped into two separate layers; transport and control layers. The AN concept assumes the presence of an IP-based layer to ensure the connectivity between different networks. Moreover, ANs extend all-IP networks by means of network composition, enhanced mobility, and support for heterogeneity.

Dynamic composition of networks is required not only at the addressing and routing levels, but it also incorporates higher layer support such as content distribution and service control functions. Network composition in this case functions across operator and technology boundaries without user involvement.

Mobility in Ambient Networking includes intra-domain mobility within a static network, roaming across domain boundaries, and integrating localized communications and device-

service interactions. Enabling QoS and optimal routing of multimedia flows requires an efficient interaction between mobility and the control interfaces.

Given that ANs are based on a federation of multiple networks with different operators and technologies, it increases the user's ability to freely select technologies and service offerings thus increasing affordability of ubiquitous communication. However, different technologies have to be integrated into the networks. Ambient Networking takes into consideration the diversity of access links, the impact of network heterogeneity on end-to-end QoS and multimedia delivery, and provides media flow routing and transport functionalities in the overlay AN architecture.

The above AN functionalities are incorporated into the Ambient Control Space (ACS) [27]. The ACS abstracts control functions of existing networks to a common set of Functional Areas (FAs), each of which is responsible for a certain value-added functionality (e.g. QoS, mobility, security, policy management, network management, or media service delivery). For operators, the AN concept allows for flexible and dynamic network configuration and management. Smart Media Routing and Transport (SMART) architecture [28] was proposed to enable the seamless integration of multimedia services into Ambient Networks.

2.1.1. Roles in SMART Architecture

In SMART a user requests a service from a content/media provider (Media Server, MS). The MS makes a request over the Ambient Service Interface (ASI) to the AN provider for a Service Specific Overlay Network (SSON). A service profile containing policies and an SLA is exchanged between the MS and AN provider. Once bootstrapping is completed, data is transported to and from the user over the established SSON. The AN provider acts as a mediator and selects a set of service nodes that perform caching, media adaptation and routing. The above complexity is hidden from the user and only interactions with the MS are visible to the client.

Services as defined in the SMART-context can be simple requests of information, multimedia streams, or can be more complex services including mobility features, media

adaptation features, caching features and so on. Processing of media traveling along a path in the network may be needed (e.g. transcoding video/audio streams to changing link properties or caching streams to support user movement). End system adaptations are of limited value. Media have to be transmitted once for each type of encoding (when dealing with transcoding) when dealing with server-side adaptations. Drawbacks of client-side adaptations include increases in complexity, cost, and resource depletions at the clients' terminals.

Many services can only be provided through network-side functions. These include caching, optimal routing to optimize QoS, broadcasting and multi-party communication, or bandwidth usage optimization. This is achievable using the SMART architecture which locates providers and network-side services. The following are examples of network-side functionalities taken into consideration: media routing, media adaptation, media splitting, synchronization, and smart caching. These multimedia transformations in SMART are performed by network-side processing services [29]. These are referred to as Media Ports (MP) and they are found on the media path between the Media Client (MC, or sink) and the Media Server (MS, or source).

2.1.2. Service Specific Overlay Networks

SSONs are defined by a set of overlay nodes and links constituting a virtual network. The inclusion of intelligent, interacting service nodes forms an SSON. These nodes are referred to as *Overlay Nodes (ONodes)*. A single ONode can provide more than one media function at a time, and thus can also belong to more than one SSON simultaneously. Overlays build virtual topologies over specific network technologies where each overlay node and link may not directly correspond to nodes and links at the network level. The underlying network can thus provide the necessary connectivity and QoS for the flows using available QoS technologies (DiffServ [30], IntServ [31]). The overlay level on the other hand contains ONodes that have the necessary states to associate individual flows to specific SSONs.

One of the important advantages of the overlay concept is that it enables the establishment of different types of overlay networks as needed. This allows, for example,

tailoring the virtual addressing scheme and the overlay routing to best suit the requirements of a particular service, and more advanced multimedia transport techniques that enable transparent integration of value-added media processing capabilities into the end-to-end media delivery path. Because of these tremendous capabilities, the overlay concept has been selected as the basic building block for the SMART framework.

In SMART a different virtual network is deployed for every media delivery service (or group of services). This permits configurations of appropriate high-level routing paths that meet the exact requirements (for example, QoS, media formats, responsiveness, cost, resilience, or security) of a media service. Moreover, the exploitation of overlay network techniques also facilitates the transparent inclusion of network-side media processing functionalities (such as caching, adaptation and synchronization) into end-to-end data paths. Besides, the overlay network is able to react dynamically to a changing environment, that is, modifications in the overlay might be triggered due to changes in user preferences, mobility, QoS or the underlying network. Finally, to provide maximum flexibility, SMART supports all these actions separately for each flow of the media service within an SSON.

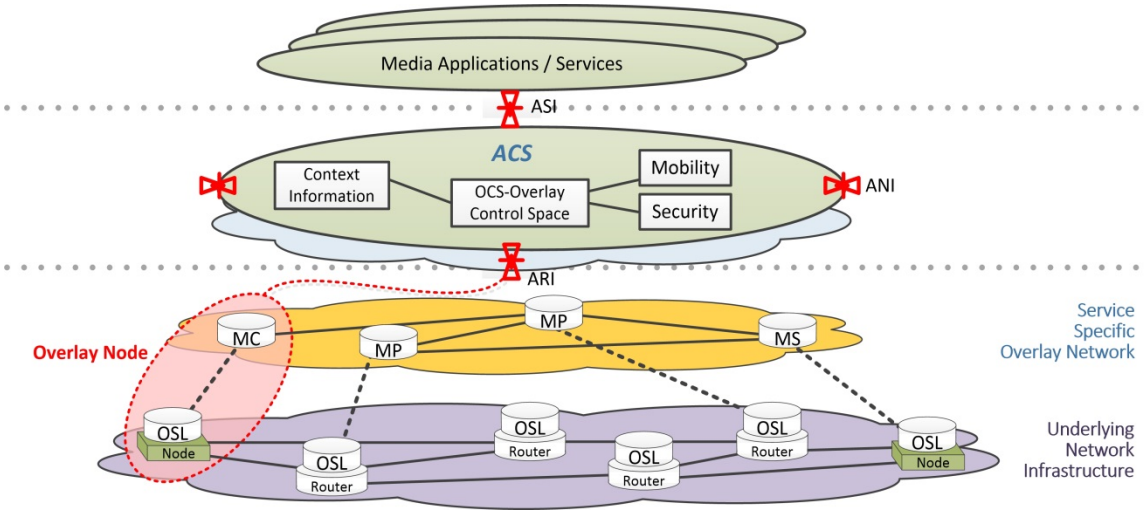


Figure 2.1 SMART architecture redrawn from [14].

Figure 2.1 (redrawn from [14]) illustrates how the SMART architecture relates to the overall AN architecture. The figure also shows the Ambient Control Space (ACS) as well as its control [Ambient Service Interface (ASI), Ambient Resource Interface (ARI)]. The ASI provides services and user profiles to the Overlay Control Space (OCS) in case of a request

for a media delivery service. The ARI is an interface to the connectivity layer and manages underlying connectivity resources.

2.1.3. Overlay Node (ONode) Architecture

An ONode is a specialized Ambient Network node that implements the functionality required to join SSONs by, for example, provisioning network-side media processing functionalities, such as caching, media adaptation, and synchronization, inside the network. ONodes (see Figure 2.2, redrawn from [28]) can be described from the user perspective and the control perspective. For each SSON of which the ONode is part, MediaPorts (MPs) are instantiated. MPs are responsible for Media Routing in the control plane and user plane. It also hosts the so-called application modules, each responsible for a particular network-side media processing functionality. Furthermore, and depending on the required media processing functionality, overlay nodes can take one or more roles of MCs, MSs, and MPs. Note that a physical ONode can be part of many SSONs at the same time.

The control plane of the ONode includes the ONode Control entity, which is responsible for the general management of the ONode and signaling exchanges. The ONode Control consists of several components, which can be classified into those that deal only with the local control and management of the ONode and those that logically belong to the OCS. The OCS is a Functional Entity residing in the ACS that controls SSONs on an Ambient Network wide basis.

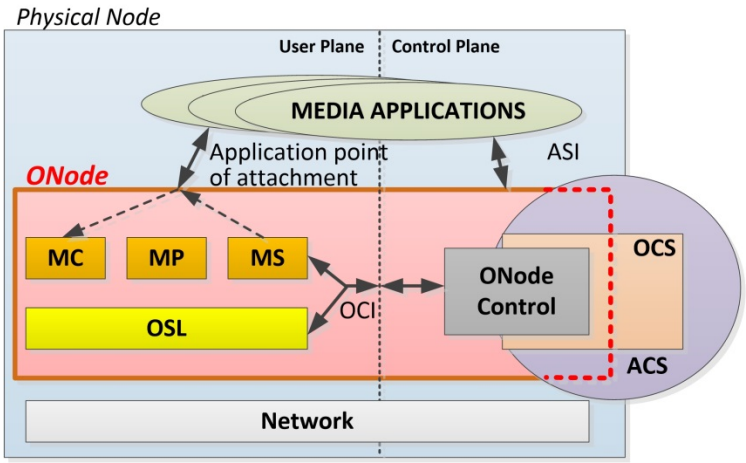


Figure 2.2 SMART ONode architecture, redrawn from [28].

The user plane of the ONode encompasses the Overlay Support Layer (OSL) and the application modules that take part in media processing actions. The OSL sits on top of the underlying network; it embodies the basic overlay network functionality required in every ONode for handling packets at the overlay level. As such, the OSL is responsible for sending, receiving and forwarding SSON-level packets. The OSL provides a common communication abstraction (overlay level network protocol and addressing) to all ONodes of an SSON, so that they can communicate with each other independently even with differences regarding the underlying protocol stacks and technologies. On top of the OSL, and using its services, there are application modules that implement the behavior of a MC, MS or MP in regard to data handling. MCs act as data sinks and send the multimedia data to end-point media applications; whereas MSs act as data sources and receive multimedia data from the end-point applications.

2.1.4. SORD

Built on the SMART architecture, SORD [32] is a fault-resilient service overlay for MediaPort resource discovery. The work presents an efficient and accurate approach to locating network-side services for SSON compositions. The authors of [32] address problems of inefficiency and large message overhead typical of traditional approaches to resource discovery. SORD is further based on optimal Chordal rings; a similar approach to our Hybrid Service Overlay Network presented in Chapter 4 of this dissertation, but with key differences. SORD considers the types of services offered by MPs as well as their physical location by providing SMART with an overlay that can be efficiently queried without using service announcements. The work illustrates that nodes without any services or those not located in the route to the desired resource are not involved in the discovery process. Given SORD's use in dynamic networks, a decentralized approach is utilized. Moreover, SORD can adapt to the changing topology with low overhead, and maintain connectivity even with frequent node failures.

Message forwarding in SORD is done within an α search scope angle. That is, any node receiving a query only forwards it to its neighbors if it falls within α. To determine whether it falls within the search scope angle, each node computes the angles β and $\emptyset$ using the

following formula:

$$\beta = sin^{-1}\left(\frac{|(x2-x1)(y1-y0)-(x1-x0)(y2-y1)|}{\sqrt{(x2-x1)^2+(y2-y1)^2}\times\sqrt{(x1-x0)^2+(y1-y0)^2}}\right) \quad (2.1)$$

Where *(x1, y1)* is the MC location, *(x2, y2)* is the MS location, and *(x0, y0)* is the location of the current MP. $\emptyset$ is computed in a similar way except for switching the location of the MC and the MS. As such, the node falls within the scope angle if: $\beta < {}^{\alpha}/_{2}$ *and* $\emptyset < 85^{\circ}$. Queries are only processed by nodes within the search scope area (See Figure 2.3). The processing involves retrieving the incomplete composition chain history, the node adding itself if it can perform the service, and then forwarding the query to its neighbors.

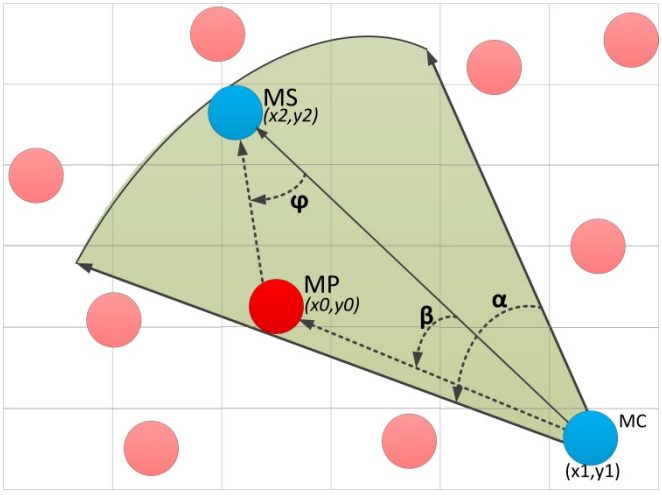


Figure 2.3 SORD scope search angle for MP discovery, redrawn from [32].

SORD provides a greater degree of freedom when it came to resource discovery for SSON composition than similar research. SORD thus forms SSON compositions based on three assumptions:

- The location of the MC is known
- The location of the MS is known
- Search for MPs is performed in the direction emanating from the MC towards the MS only.

The first two assumptions are necessary and we agree completely with the need to have such points of reference when constructing SSONs. However, in wireless mobile networks, the dynamic movement of nodes at the network layer may result in an unequal distribution of MPs at the overlay level. We may end up with high density regions where large numbers of diverse MPs exist while other regions may lack any MPs or have them sparsely located. In the latter case, it is highly likely that α must be continuously increased until all necessary component services are found. This constraint of the above solution results in a significant increase in the number of query messages needed to find a valid MP with both, the required service functionality and the QoS level needed to meet the MC's expectations.

In our work, we provide an alternative to the previous solution using two methods. This is in reference to our plan-based composition only and not the plan-free solution. The latter is characterized with even greater flexibility and decentralization. First is by providing a hybrid service overlay topology discussed further in Chapter 4. Second is by the dynamic node promotion and demotion used in the hybrid topology. These two methods result in what can be described as a circular MP space that is further re-organized based on the types of services provided. As a result, nodes arriving in the network must first determine the overlay region they must join. The resultant is an overlay where searches can be done within precise areas of the overlay for specific types of services with specific quality levels.

The use of the hybrid service overlay network abstracts the geographic position of MPs, and routing between nodes is reduced to locating links in the hybrid overlay rather than locating network layer links. One disadvantage of our method is that reaching some MPs may not be as sufficient as SORD. This is due to the fact that some nodes that are neighbors in the overlay layer may in fact be geographically more distant. However, three advantages are brought on by our H-SON. First, through overlays we can now guarantee that any MP can now be reached with composition queries regardless of their geographic location. Second, discovering MPs with specific types of services and QoS levels can be done by directing queries to bounded regions of the H-SON. Third, any service query sent will most definitely result in a successful discovery of at least one MP that meets the functional and non-functional requirements. If no MP is found then it can be concluded that no such MP exists anywhere in the network and the service requirements have to be adjusted to find alternative MPs with lower quality levels. In SORD however, service discovery failure

results in an increase in the search angle α and a reattempt in sending the service query. This process continues until either an MP matching the requirements is found, or a failure is declared. As such, our approach is generally faster in discovering services, requires a smaller number of query messages, and can guarantee an SSON composition that more accurately meets the MC's QoS requirements.

2.2. Overlay Networks

Overlays are virtual networks formed by logical nodes and links that are built on top of an existing network. Each overlay link is usually composed of one or more physical links. The goal of overlay networks is to implement services not available in the existing network such as increasing security, routing robustness, etc. The Internet as we know it is an overlay network built on local area connections (e.g. Ethernet) and phone lines whose goal is to connect local area networks. For the Internet to function, an Internet protocol header is added to all packets. Implementing overlay networks does not require the deployment of new equipment, or the modification of existing software and protocols. The need, however, might arise in deploying new software on already existing ones. In addition, overlay networks do not have to be deployed at every node in the network. This allows nodes not interested in joining an overlay to avoid becoming part of the overlay network. This decision might stem from the nodes' lack of resources to support heavyweight overlays, the unclear security properties of overlays, or the scalability issues associated with some of the overlay topologies (e.g. requiring n^2 states or communication). There is however some costs associated with overlay networks. The addition of a layer in the networking stack, thus adding packet headers introduces additional overhead and processing requirements. Furthermore, layering does not eliminate complexity; it may only possibly manage it. Consequently, the addition of more layers of functionality may introduce unintended interaction between the layers.

Frameworks developed for this purpose fall mainly into one of two configurations: *static* and *dynamic*. They can be further classified into *application-specific overlay networks* and *generic overlays* supporting diverse applications. Moreover, overlays can be classified according to their topologies as *structured*, *unstructured*, or a *hybrid* intermixing of the two.

The need to provide a brief discussion regarding overlay networks stems from our first solution's utilization of a service overlay network onto which SSONs are composed. Our overlay falls under the dynamic, generic, hybrid overlay categories.

2.2.1. Application-Specific and Generic Overlay Networks

Overlay networks designed and tailored specifically for certain applications are referred to as application-specific overlays. These include peer-to-peer file sharing, end host multicasting, and content distribution networks. Such applications require an appropriate support in terms of topology discovery, routing path selection, and fault detection and tolerance. Overlay applications rely on end hosts [33] [34], some of which are preset because of the limited access bandwidth and unpredictable connectivity of many hosts [35] [36] [37]. These nodes have the responsibility of detecting changes in network status, discovering the topology, providing fault tolerance, and guaranteeing QoS. However, with the increasing number of applications available, more and more of these dedicated nodes are required. Much of the widely used P2P and overlay networks have been tailored for specific applications. Chord [38] and Bayuex [39] were designed for well-defined-structure-based overlay applications whose goal is to locate content in large distributed systems.

Similarly, RON (Resilient Overlay Networks) was an application-layer overlay on top of the existing Internet routing substrate [11]. RON improved recovery from path outages and periods of degraded performance by allowing small groups of distributed Internet applications to detect and recover from said situations. Ultimately, RON's main goal was to develop a technique that allowed end hosts and applications to gain improved reliability and performance in the Internet through cooperation. Each RON node monitored the quality of adjacent Internet paths and used that information to determine whether packets had to be routed through different paths over alternate RON nodes.

RMX (Reliable Multicast proXies) agents split large heterogeneous reliable multicast sessions into several multicast data groups of co-located homogeneous participants [40]. The groups were organized into spanning trees using an overlay network of TCP connections. Each RMX was responsible for forwarding data generated from one local

group to the rest of the data groups. This work illustrated how applications could dynamically adapt to the inherent heterogeneity of the Internet by acting as an intermediary between sources and consumers and dynamically adapting content and data rate to best suit the clients' needs and interests.

Overcast [41] was an application-level multicasting overlay that was incrementally deployed over the Internet to provide scalable and reliable single-source multicasting using a simple protocol for building efficient data distribution trees adaptable to network conditions. Overcast overlays work well on large-scale networks supporting multicast groups of up to 12,000 members. Malouch et al. [42] introduced an algorithm with the utilization of proxies to meet fixed delay bounds while minimizing costs. The algorithm's optimality was proved in a fully-connected overlay network with uniform length edges. The solution used proxies as multicast forwarding points. These proxies were high-bandwidth multicast forwarding devices that were provided at a cost. Furthermore, a tunable heuristic was used where delays between application layer multicast points were not uniform.

Another application-level multicasting overlay that scaled to arbitrarily large receiver groups while tolerating failures in routers and network links was Bayeux [43]. The overlay relied on the Tapestry application-level routing protocol [44] and included mechanisms for balancing loads across replicate root nodes and resulted in efficient bandwidth consumption. Furthermore, Bayeux combined randomness for load balancing with locality for efficient use of network bandwidth. More importantly, Bayeux provided a 'first reachable link selection' protocol to leverage the redundant routing structure of Tapestry.

The SON as a partial component of our work differs from the above in that it utilizes an overlay as a base onto which services can be composed efficiently. Thus a number of overlays are constructed and managed at the same time over the provided overlay. Furthermore, no dependencies exist between SON nodes regarding shared content (such as the case in P2P networks) but rather attempts to autonomously maintain a stable overly in a dynamic environment.

In generic overlay networks, knowledge is shared through an intermediate layer that measures a number of network properties over which further overlays and/or applications can be constructed. Planet-lab [45] [46] is one of the most famous examples; it is an

experimental global research network that supports the development of new network services such as distributed storage, network mappings, P2P systems, distributed hash tables, and query processing.

Nakao et al. [47] presented a routing underlay for overlay networks. The routing underlay sat between overlay networks and the underlying Internet. The underlay was layered thus permitting specialized routing services to be built above it from a set of basic primitives. The underlay exposed large-scale, coarse-grained static information collected by the network, and upper layers were able to perform frequent probes over a limited set of nodes rather than the whole network. The underlay extracted and aggregated topology information from the underlying Internet and supported three primitives: first it provided a graph of the known network connectivity at specified resolutions and scopes. Second it exposed the paths taken by packets from one point to another. Third it reported topological facts about specific paths between any pair of points.

Saxon was a distributed software layer that dynamically maintained a selected set of overlay links for groups of nodes [48]. The software's main objectives were to provide an easy-to-use software layer with a wide range of applicability that reduces latency, overlay hop-count distance and bandwidth use. The target services for Saxon were short and long unicast, short and long multicast/broadcast, and periodic neighbor nodes communication.

Opus [49] a large-scale overlay utility service provided a common platform and abstractions for simultaneous hosting of multiple distributed applications. Opus allocates available nodes to meet the requirements of competing applications based on observing dynamically changing system characteristics and application access patterns.

X-Bone [50] was a system for the rapid deployment and management of multiple overlay networks. It consisted of existing overlay technologies combined with management tools and coordination services. X-Bone provided a configurable virtual networking infrastructure useful in developing network and application services and useful for deploying isolated infrastructures. X-Bone provided isolation between overlay networks through partitioning of resources. Through this, experimental overlay networks could be built over X-Bone without any interference.

Another example of generic overlays is OverQoS [10]. It was an overlay-based

architecture for enhancing the best effort service of the Internet. OverQoS provided several services such as smoothing packet losses in bursty networks, prioritizing packets within an aggregated stream, and providing statistical loss and bandwidth guarantees. The placement of nodes in OverQoS was pre-specified with nodes spanning different routing domains. Loss control is achieved through Controlled-loss Virtual Links (CLVLs) abstraction. It provided a bound on the loss rate seen by the bundles over a certain period of time. Overlays were able to achieve this loss bound by recovering from network losses through a combination of Forward Error Correction (FEC) and packet retransmissions.

A last example is the Service Overlay Networks (SON) [13]. It was first introduced as an effective means to address some of the end-to-end QoS issues plaguing the Internet. SONs also facilitate the creation and deployment of value-added QoS-sensitive services. This is done by purchasing bandwidth with certain QoS guarantees via service-level agreements (SLAs) to build virtual end-to-end (E2E) service delivery infrastructure on top of the existing data transport networks. SLAs are well-defined business relationships between the SON, the underlying network domains and users of the SON. SONs decouple application services from network services, thereby reducing the complexity of network management and control.

Generic overlay networks discussed above have been able to efficiently reduce the cost associated with acquiring knowledge and increasing the E2E QoS. However they do not take into account user mobility and the specific demands for individual services. The majority of generic overlays do not provide dynamic methods for overlay configuration or an efficient approach in locating specific network-side services provided by member nodes. Our hybrid SON utilizes node services, quality levels, dynamicity and stability to build a generic SON that can enhance the construction of SSONs according to pre-specified requirements for new service deployment for incoming users.

2.2.2. Structured, Unstructured and Hybrid Overlay Networks

Overlay networks can be classified as structured or unstructured. Structured overlays impose constraints on the network topology for efficient resource discovery, while

unstructured overlays organize nodes in a random topology that is more resilient to peer stability and dynamicity [51] [52] [53] [54]. Dividing overlay networks into structured and unstructured topologies has been a key component of Peer to Peer systems (i.e. application-specific overlay networks). Unstructured overlays have large flexibility in selecting their neighbors and in their routing mechanisms. Moreover, unstructured overlays can create topologies that are resilient to node failures. Unstructured P2P systems are built by establishing random neighboring relations across nodes. Most importantly, unstructured overlays have low costs to build and maintain; a cost that is shared among all nodes in the system. Unstructured overlays usually present some level of redundancy thus making them more resilient to node failures and message losses. Random unstructured overlays have several properties. First is their high connectivity degree thus permitting every node to reach every other node. Second is their uniform degree of edge distribution. This measures the reachability and contribution of each node to maintain the overlay. Third is their low clustering coefficient; a value indicating the number of edges between the node's neighbors divided by the maximum possible number of edges across those neighbors.

Gnutella [55] is one of the most popular P2P file sharing unstructured overlay designs. It utilizes a 'ping' maintenance message to discover hosts on the network. Nodes reply with 'pongs' containing information about the responding peers and other nodes they are familiar with. Queries in Gnutella (early versions) [56] employed a simple flooding strategy. Queries were propagated to all neighbors within a distance of a certain number of hops. A newer version of Gnutella [57] organizes the overlay into a two-level hierarchy that distinguishes between superpeers and leaf-peers. The network core consists of high capacity superpeers that connect to other superpeers and leaf-peers. The second level consists of low-capacity leaf peers that perform low priority tasks if any.

Another unstructured P2P overlay architecture is eDonkey [58]. It is one of the most popular P2P file-sharing networks and utilizes a hybrid architecture consisting of two tiers. The first tier is composed of servers that index files. The second tier consists of clients downloading and uploading the files. Clients themselves are also divided into two groups: The first group consists of clients that can accept incoming connections and are given high IDs. The second group consists of clients that cannot accept connections and are given low IDs. Maintaining stability in eDonkey is done by servers adding clients that frequently send

connections or search requests to servers into blacklists and refuse any communication with them. After a predetermined period of time, these clients are removed from the blacklists to resume communication. Clients in eDonkey can download a file by gathering a list of all potential file providers. This is done by sending a query to one of the servers. Each server keeps a list of matching files and returns their location to the client.

Another P2P protocol that was used by Kazaa and Grokster [59] [60], is FastTrack [61]. It was one of the most popular file sharing protocols with over 2 million users. FastTrack employed the UUHash hashing algorithm which allowed checksumming of large files in a short period, however it permitted massive corruption files to leak unnoticed.

The problem with unstructured overlays is their limitation to unguided search approaches that use either flooding or random walk. Such methods cover large numbers of nodes yet introduce a large amount of unwanted traffic. Consequently, network bandwidth is inefficiently wasted.

Structured P2P overlay networks on the other hand introduce a tighter control on the overlay structure, routing protocol, and node placement. Each node is assigned a unique ID (through hashing) and maintains a routing table containing *O(log(N))* entries, where *N* is the total number of nodes in the overlay. Usually, the location of a data object is a function of the object's hash value and a peer's ID. Consequently, the DHT-based systems can locate an object within a logarithmic number of steps using only *O(log(N))* query messages. Examples of structured overlays include Overnet [62], Kademlia [63], and Chord [38]. Overnet was a closed-source protocol that relied on Kademlia as its underlying DHT protocol. Overnet assigned each node a 128-bit ID based on the MD4 hash. Each object is assigned an equal-length ID.

Kademlia assigns a 160-bit hash ID to each node and computes a 160-bit equal –length hash key for each data object by hashing the content using SHA-1. Key-value pairs are placed on nodes with IDs close to the key. Moreover, each node builds a routing table consisting of up to $log_2(N)$ buckets, with the *i*th bucket B_i containing IDs of peers that share an *i*-bit long prefix. Similarly, *N* refers to the current number of nodes in the overlay. The placement of node entries in Kademlia buckets is flexible. Additionally, Kademlia supports efficient lookup for the *k* closest peers for any given node key. The implementation details

of Kademlia's buckets guarantee a consistent *O(log(N))* upper bound.

Chord is characterized by its simplicity, provable correctness, and performance. Simplicity stems from its ability to route a key through a sequence of *O(log N)* other nodes towards the destination. For routing efficiency, each node must keep information about *O(log N)* other nodes. However, performance degradation is slow when this information becomes out of date. Such ability is crucial in an environment where nodes can join and leave arbitrarily. Correct routing in Chord is guaranteed even if only one of the *O(log N)* nodes' information is correct, though performance is affected. Some of the other benefits of Chord include the fact that it does not impose any naming structure and can find data objects that are not tied to any particular machines. Although Chord does not provide anonymity, however lookups are performed in predictable time. Furthermore, Chord achieves scalability without involving any hierarchy. Some of the difficult problems tackled by Chord include load balance, decentralization, scalability, availability, and flexible naming.

Balance is achieved through an even distribution of keys over nodes, thus balancing computation loads. Given that all nodes are treated equally in Chord, this enforces its decentralization and improves its robustness. The cost of a Chord lookup grows as the log of the number of nodes, thus it has a high level of scalability. The automatic update of internal tables to reflect node joins and failures, the node responsible for a specific key can always be found. Finally, Chord has no constraints on the structure of the key space providing large flexibility in this field.

When a new node joins Chord, only *O(1/N)* fraction of the keys are moved to a different location. In an *N*-node network, each node maintains information about *O(log N)* other nodes, and any lookup requires only *O(log N)* messages. Furthermore, joining or leaving a Chord overlay requires $O(\log^2 N)$.

Placement of nodes in Chord depends on the node's IP address and a key identifier generated by hashing the key. The length of a node identifier is *m* bits long; a length that must insure two nodes or keys hashing to the same identifier is a negligible probability. Identifiers are placed in an identifier circle $modulo\ 2^m$, and key *k* is assigned to the first node whose identifier is equal to or follows *k* in the identifier space. This is referred to as the successor of *k*, *successor(k)*, the first node clockwise from *k*. When a new node joins the

network, certain keys previously assigned to *n*'s successor are now assigned to *n*. Once *n* leaves, those keys are reassigned to its successor.

2.3. H-SON Related Work

Guo et al. [64] presented a hybrid Peer-to-Peer (P2P) Overlay network for highly efficient searches utilizing a simple approach in dividing the overlay into structured super peers and standard unstructured peers. Super peers are connected through a ring-like structure, while each super peer forms an optimal tree of ordinary peers connected to it. Super peers are chosen based on the evaluation of their connectivity degree compared to a threshold value. Queries in the hybrid structure have a maximum number of hops which if not satisfied then the query is dropped. A major drawback of the above approach is the inverse relationship between the overlay's re-configurability and resource availability seen in the breadth and depth of the optimal tree each of these super nodes forms. If each super node has an unlimited bandwidth connection and unlimited resources, then an unlimited number of regular peers can be managed by a super peer. However, in real networks, resource limitations in super peers require placing limitations on the number of direct regular peers connected as well as how deep trees can grow. A failure of a high-level peer in one tree may require costly reconfigurations. Additionally, the above paper did not illustrate how a failure should be handled to minimize information loss, the message overhead for network reconfiguration and stabilization.

Li et al. [65] presented a solution to achieve a sustained throughput and fast data dissemination rate from any data source with a goal of minimizing overhead even under frequent node failures through a Churn-Resilient Protocol (CRP). The Chord-like structure proposed dynamic fingers among nodes satisfying certain proximity properties for fast data dissemination. From the CRP overlay, a shared spanning tree is extracted with minimized latency. In the initial hops of the tree, scope-flooding is used and tree traversing is used in the remaining hops. The authors correctly indicated that tree-based dissemination methods are vulnerable to node failures. Consequently their solution proposed the use of churn resilient overlay nodes to achieve good fault resilience. New nodes form finger links to

other ring nodes based on proximities. Proximity is measured using latency and capacity thus placing high-capacity nodes at high positions in the delivery tree.

The purpose of choosing a Chord-like structure is to overcome the vulnerability of tree structures to node failures. Any node failure can be temporarily bypassed through links neighboring nodes have to other Chord nodes in the overlay. Hence if any node fails with probability e^{-a}, where $a > 1$, then the structure is connected with a probability higher than $1-n^{1-a}$. A link between any two nodes is given a network proximity weight that is measured by the link's latency and the capacity difference between the two nodes. Although the solution improved traditional Chord network, it's reliance on structured P2P overlay meant data placement and topology of the network are tightly controlled. Although structured P2P solutions have efficient data access and reduced number of search hops, processing fuzzy queries is challenging. In addition, the stabilization costs associated with the arrival and departure of nodes exceeds the original cost of Chord.

To reduce search time for available services in P2P systems, Laskowski [66] illustrated a super-chunk-based fast search network (SURFNet) for prompt Video on Demand (VoD) delivery. Their design had some limitations: the need for timely content location and delivery, and limited cache size. Therefore, multimedia files are divided into chunks and grouped to form super-chunks. Peers form a structured overlay based on the super-chunks they hold. This structure consists of an AVL tree layer and a holder chain layer. The former is formed by stable peers that are online for a long time and provide a stable backbone. The latter is a linked list grouping all peers holding the same super-chunk together and attached to a stable peer in the AVL tree. All remaining peers form an unstructured overlay where they periodically exchange chunk-level data availability information with neighboring nodes.

A work presented by Makaya et al. [67] focused on user-generated contents and user-generated services in Next-Generation SONs (NGSON). The objective of the work was to support efficient delivery of services and applications while being end-user-centric by allowing network operators and service providers to have service management, control, creation, composition, and execution deployment. However, the approach used by the authors was highly centralized requiring an NGSON controller to act as an overlay manager.

The controller formed a primary point of contact for end-users and service nodes that were responsible for service composition, orchestrating, chaining, service node discovery and monitoring, overlay management for self-organization and sessions and path establishment between service nodes. However, the approach faced the problem of a central point of failure, lack of scalability, and requiring the manager to monitor arrival and departure of all service nodes. Our work focuses on finding a decentralized solution for service composition that maximizes the benefits and minimizes the limitations found in centralized and decentralized solutions.

We focus our work, in the plan-based service composition solution, on dividing the Overlay Network layer in the SMART SSON architecture into two layers; a lower overlay and the SSONs built over it. The lower overlay constitutes a hybrid overlay structure whose node organization and links reflect the type of services provided by nodes, their stability and QoS levels.

The lower overlay network is composed of all MP nodes present that can be utilized during the formation of a service composition between the MS and the MC. MPs are chosen to form SSONs reflecting the MC's needs in terms of QoS and expected QoE. By using a lower overlay network, we are able to organize MPs prior to any media and service requests from MCs for fast MP search and SSON construction. Nodes are assumed to be mobile, hence node failures are expected. We use a hybrid lower overlay network for two reasons: 1) to reduce the number of messages needed to find component services with specific QoS levels without flooding the network with unnecessary messages. 2) to restrict MP queries in the direct path from the MS to the MC thus decreasing the hop count bot at the network and the overlay layers.

Our to-be presented hybrid structure forming the lower overlay network is used to overcome limitations seen in both structured and unstructured overlay networks. Unstructured overlay networks are formed with arbitrary overlay links. These networks are easily constructed. However, there is notable limitation when a peer is interested in acquiring some data. In this case, the query needs to be flooded in the network to find a node holding the desired data. The less popular the service the client is interested in acquiring, the less likely it will be found in a nearby node to the requestor. Consequently,

the number of messages exchanged would consume significant bandwidth in a wireless network besides the operational overhead. In addition, since there is no correlation between the service requested and the corresponding MP, it is highly unlikely that a search is successful.

Structured overlay networks overcome many of the limitations of unstructured networks by maintaining DHTs and distributing the content in the network. The disadvantage of structured overlays is the large number of messages that need to be exchanged to stabilize the network, maintain existing links, and heal links to failing nodes that drop from the network unexpectedly. This rigidity is highly unfavorable in dynamic mobile environments where nodes leave the network without warning and new nodes are continuously arriving.

Hence, we propose the use of a hybrid overlay structure that maximizes the advantages of both structured and unstructured overlays and minimizes their disadvantages. Our overlay structure consists of several hierarchical layers of Chord-like rings and unstructured leaf connections. The details of this service overlay network are discussed in Chapter 4.

Chapter 3

State of the Art in Service Discovery and Composition

The previous chapter introduced an overview of existing work regarding overlay networks in general. We illustrated the need for a generic, hybrid, and dynamic service overlay network onto which service-specific overlays can be composed. This chapter discusses some of the related work that exists in terms of service composition languages, architectures, and service discovery methods. Section 3.1 defines service composition and presents some of the existing composition languages. Section 3.2 discusses some of the centralized and decentralized service composition architectures. Section 3.3 discusses probing-based and hop-by-hop service discovery. Section 3.4 presents related work in the field of semantic-based service discovery. Finally, Section 3.5 discusses autonomic service discovery and composition in mobile environments.

3.1. Service Composition Languages

Service composition allows simple services to be dynamically combined into new, more complex services. Initially composition was restricted to web services (mostly applications), but eventually extended to network-side functions such as media modification and synchronization. A large number of languages and techniques for service composition have emerged and are continuously evolving. A number of XML-based standards to formalize the specification of services, web services to be precise, have been developed. Similarly other languages for service composition and execution have also been presented in the

literature. In the following subsections, some of the existing service definition and composition languages are presented.

3.1.1. BPEL and BPEL4WS

WS-BPEL, or BPEL [68] [69] (Web Service Business Process Execution Language) is a standard executable language for specifying actions within business processes with web services. BPEL defines a model for describing a business process as a set of activities. The modeling process using BPEL can be time-consuming and error-prone, hence the large reuse of existent models. The use of BPEL can facilitate the implementation of service compositions. BPEL is considered to be an orchestration language that specifies the executable process that involves message exchanges with other systems.

BPEL4WS [70] is an XML-based specification language for specifying business processes exclusively based on Web services. It defines how multiple services can interact through rules of coordination via Web services interface, necessary to achieve business goals. It supports the definition of both executable business processes and abstract business processes by providing mechanisms to specify common core concepts of both types of processes with essential extensions for each process.

BPEL4WS employs partner links to directly model peer-to-peer collaborations between business processes and partners services. When messages or events are received from partner links, an instance of a business process can be created. Partner links define static and abstract relations with other partners based on Web services *portType*s used in interactions. To dynamically select a service provider for particular types of services, BPEL4WS uses the notation of endpoint reference. Moreover, BPEL4WS binds web services into cohesive units encapsulated in activities. However, there are a few limitations associated with BPEL and BPEL4WS. Its weak representation of roles involved in the business process. Role representations are sometimes necessary to reflect the responsibilities and behaviors of the parties involved in various scenarios. Moreover, BPEL does not have semantic support. This is necessary in composition languages to facilitate the automatic composition of services. In addition, semantics enable the automatic discovery and invocation of services in

an autonomic system. Most importantly, BPEL does not support business agreement. In our presented situation, an agreement represents the non-functional requirements of clients, i.e. required QoS and media stream description (encoding, etc…).

3.1.2. WSCI

WSCI (Web Service Choreography) [71] [72] was an XML-based language built on top of WSDL (Web service definition language) that provided a standard for specifying the overall collaboration between service providers and users through request and response messages. It builds on the WSDL *portType* capabilities to describe the flow of messages exchanged by a service within a process. Moreover, WSCI permits the description of the external behavior of a service thus facilitating expressing sequential and logical dependencies. However, WSCI lacks modeling and execution control abilities. These are the ability to assemble and incorporate individual services into an execution process that is necessary for any service composition language. Similar to BPEL4WS, WSCI also lacks semantic support and business agreement support.

3.1.3. WS-CDL

WS-CDL [73] is an XML specification used to compose interoperable, long running, P2P collaborations between service participants. A global view of exchanged messages and behavior of all service participants involved in the business collaboration is provided by WS-CDL. WS-CDL is independent from the platforms and programming languages used to implement the participating services. Peer-to-peer collaboration between participants in WS-CDL are modeled with different roles using Choreography. This utilizes Interaction and Activities notation to define the interaction and message exchange between two services participants described in WSDL. Synchronization of activities can be achieved through *WorkUnit*, which defines the guard condition that must be fulfilled to continue the activity. WS-CDL lacks that same requirements as those of WSCI.

3.1.4. DAML-S

DAML-S [20] is an ontology-based service language to establish a framework within which web services can be described in the context of the semantic web. It is a DAML+OIL ontology for describing the properties and capabilities of web services. DAML-S has the goal of making web services computer-interpretable and to enable the following tasks [74]:

- Discovering and locating services that provide a particular service adhering to specified constraints. Locating services is usually performed through a registry service.
- Invocating/activating and executing the identified service through agents or other services.
- Interoperation through semantics. Automatic insertion of message parameter translations between clients and services.
- Composition of new services through automatic selection, composition, and interoperation of existing services.
- Verification of service properties.
- Execution of the composition and monitoring the execution.

The semantic description of the service in DAML-S is divided into three parts: service profile, process model and grounding. The service profile describes functionalities of the service by specifying the input and output types, preconditions and effects. The process model describes the service function. It describes how the service works as either an AtomicProcess or a CompositeProcess. The former is a simple process that is executed directly, while the latter is a combination of two or more subprocesses. Finally, the grounding contains details on how services are accessed through communication protocols, protocol parameters and serialization techniques. DAML-S has the ability to express the entities using the concepts defined in semantic web ontologies thus providing expressive constructs suitable for the automatic discovery and composition of services.

3.2. Service Composition Architectures

Research into service composition has evolved rapidly over the last decade. Much work has concentrated on web services, eventually leading to network-side services and the composition of SSONs. Initially, service composition architectures were designed for static environments where nodes were assumed to lack any type of mobility. As such, most architectures relied on a central coordinator for querying and composing services. However, most of these designs presented a problem when applied to highly dynamic environments where no natural central control point existed. Attempts to address these issues were researched using decentralized service composition architectures. In the following subsections we discuss some of the existing works regarding centralized and decentralized service compositions.

3.2.1 Centralized Service Composition

Fujii et al. [75] presented a semantics-based service composition architecture. It intuitively obtained the semantics of the requested service. The required components were then discovered and composed into a service based on the service's own semantics and the semantics of necessary components through a service composition mechanism named Semantic Graph-Based Service Composition (SeGSeC). The architecture composed a service by creating a workflow of the requested service using discovered components and executing the workflow.

CoRE (Component Runtime Environment) was the central entity in [76] consisting of two interfaces: the discovery interface and the access interface. The former provided an interface to discover components distributed in a network. The latter provided an interface to invoke component operations and retrieve their properties. However, the presented CoRE implementation included a discovery engine that discovered component services stored in a centralized local host, making the architecture highly centralized. This work was later extended [77] through composing an application by combining components based on their semantics and the users' contexts. However, the reliance on a centralized discovery

manager, and the separation of the composition and discovery stages, led to a longer adaptation process unfit for dynamic environments.

A QoS-based service discovery and composition mechanism was introduced by Ardagna and Pernici in [78]. The work presented a modeling approach to the web service selection problem for QoS-sensitive large processes. The service selection problem was formalized as a mixed integer linear programming problem with negotiation techniques to identify a feasible solution to the problem. The design allowed constraints on quality requirements to be specified at the local and global level. In general, local approaches can guarantee local QoS constraints. Services are selected one at a time by associating the running abstract activity to the best candidate service which supports its execution. Global approaches on the other hand allow constraints on quality requirements for the whole composed service execution. They identify a set of services that satisfy the process constrains and user preferences for the whole application. The limitation of this architecture once again is its reliance on a central registry from which services are selected.

A SOA-based middleware to support QoS control of reconfigurable service composition and energy-efficient mobile computing was introduced in [79]. The approach consisted of two parts. First, was a QoS adaptation of the client device. Second, was an adaptation of the service composition on the service-overlay network. However, all services are collected in specified directories and the steps of service discovery and composition have been separated from each other. Consequently, this is reflected through a large increase in composition delay.

Kalasapur et al. [80] presented a dynamic service composition architecture for pervasive computing. A service-oriented middleware platform called Pervasive Information Communities Organization (PICO) was employed to model and represent resources as services. PICO was employed to provide methods to extract the capabilities of resources and allow services to be built around the capabilities. However, the design was based on a service-oriented architecture that stored discovered services in a centralized directory. Users accessed the directory to locate one or more services. The directory performs a query on registered services and returns a matching service if found.

The reliance on central directories and service composition architectures such as those presented above in mobile environments is highly unfavorable. The continuous movement of service nodes quickly renders any directory content outdated. Large amounts of update messages need to be exchanged to maintain the validity of such directories. Moreover, directories represent a bottleneck for all service discoveries and compositions, and potential single points of failure. Consequently, focus has been shifted to decentralized service composition architectures such as those discussed in Section 3.2.1.

3.2.2. Decentralized Service Composition

Services in the environment may be fixed services embedded in the environment or dynamic services provided by other mobile devices. Most service composition systems have adopted a centralized approach for service discovery and coordination. However, some recent work has shifted this paradigm to partially or completely decentralized systems.

Canal [81] was a fully decentralized service composition system based on the JXTA P2P system and mobile agent concept. Service discovery, coordination, and maintenance were all built on top of the P2P system, thus avoiding the drawbacks of centralized approaches. Agents played the roles of brokers linking services together to meet user needs. The operation of Canal relied on forming service chains formed by discovering services using the JXTA P2P network. A mobile agent is sent traveling through all services in a designated order setting up the inputs and outputs. The agent returns to the client along the reverse path, triggering each of the selected services on its way.

Each service in Canal can discover and communicate with other services using JXTA protocols. However service discovery is similar to peer discovery in JXTA where two types of advertisements have to be exchanged: peer advertisements which represent the service location, and the module-class advertisements which give the service description. This method relies on an advertisement-based scheme, thus expressing knowledge about the location of specific service nodes is only acquired from surrounding peers. Furthermore, the proposed system lacks a clear method for efficiently searching for service nodes according to the types of provided services or the quality level of such services.

Rudder, a decentralized agent framework for dynamic composition of services was presented in [82]. Li et al. also relied on a P2P agent framework in decentralized distributed environments. Software agents provide high-level mechanisms for dealing with system adaptations and information discovery. The objective of Rudder was to enable runtime composition and coordination in P2P environments. The framework consisted of software agents, their interaction and negotiation protocols. These protocols enabled the behaviors of individual agents to progress towards a consensus. Adaptation plans were negotiated, decided, and enacted upon by multiple distributed cooperating agents. Two types of agents were used: Component Agents (CA) represent discrete elements (i.e. computational components or resource components), and Composition Agents (CSA). CAs are only responsible for computations performed locally within an individual element and its respective interactions. The presented work however did not provide a clear decentralized approach for service discovery. The authors' discussion suggested that elements in the network had to be registered, as such, a centralized registry system is implied.

MARE (Medium Access through Reservation) [83], an alternative MAC protocol based on reservations is presented by Gallardo et al. It provides collision-free data transmissions designed as a generalization of the RTS/CTS handshake [84]. Stations using this protocol and that are near both the transmitter and receiver, set their NAV and refrain from transmitting during specified periods. MARE introduces reservation control frames similar to RTS. Moreover, all nodes have to keep track of available slots by storing information related to all active reservations. A limit is established on the total fraction of bandwidth that can be reserved. This approach allows nodes to keep track of all reservations that can interfere with their own transmissions. Consequently, only two-hop neighbors of the data transmitter within range of the receiver are notified of the reservation. MARE permits contention-based data transmission during unreserved time intervals, which is restorable when a reservation is not needed. The authors successfully illustrated that the overhead incurred by the protocol is much lower than that of MDA.

3.3. Probe-Based and Hop-By-Hop Service Searches

The emergence of dynamic ad hoc computing environments has moved composition approaches towards decentralization. Decentralized service composition analyses the different methods suitable for locating suitable service providers without need for centralized repositories or coordination entities. Existing solutions can be divided into two directions. The first is probe-based service searches in overlay networks. The second is hop-by-hop based service selection.

Probing relies on flooding overlay networks with probe messages to search for multiple high-quality paths. A major disadvantage of the probing approach is its need for a completed composition before service execution, it postpones the finalized path selection until permitted by the requestor node. Multiple probes are usually sent to find service path candidates. Samimi et al. [85] addressed the problem of mapping services onto an overlay network through a distributed algorithm called Dynamis. The proposed method realized a generic optimization technique based on distributed selection. The technique was applied to existing probing protocols to reduce probing costs. A key property of Dynamis was that rather than performing selection only at one end point, the selection was distributed. This was done in a way that overlay nodes only forwarded a probe if it described a partial service path of significantly better quality than the quality of comparable service paths described by previously forwarded probes. Some significant reductions in overhead were observed. However, the approach required the candidate paths to report back to the source node where a path decision was made according to delay and QoS levels. In mobile environments, any delay in decision-making puts the composition at risk of being invalidated. The time it takes for candidate paths to form and report back to the source node carries with it the possibility one or more of the involved nodes can leave the network during that period. However, if resources at probed nodes are preserved for potential composition, then another problem arises in the event concurrent composition probes were received. In such case subsequent probes may be rejected due to decline in available resources at the MP even when some of the received path probes could be eventually rejected by the source node.

Park et al. [79] introduced a SOA-based middleware to support QoS control of mobile applications and to configure energy-efficient service composition graphs. A service routing

algorithm was introduced to balance power consumption on a service overlay network. Probing agents were integrated into the discovery protocol similar to that of Gnutella's pure P2P methods. Each candidate route was associated with a probing agent. A source node initiated and distributed probing agents to connected service nodes. Context information regarding response time and contention rate were updated. Then the probe agent was replicated and distributed to subsequent service nodes according to a connection-degree set by the protocol. The work presented was mostly concerned with resource depletion at the destination, represented by a mobile host. However the reliance on a P2P overlay, and the authors' indication that a path was formed through forward increments of hops from the source towards the destination suggested the presence of two assumptions. First, the existence of highly interconnected P2P networks resulting in a very high probability of direct links between each hop of the composition path. Second, the reliance on a P2P overlay structure indicating the presence of a method after which the overlay was organized according to the nodes' contents (services), something that was not discussed in the paper.

Hop-by-hop service provider selection searches for the most suitable provider in each hop. A single high quality solution is discovered for a complex request. Wang et al. presented sFlow, a fully distributed algorithm executed on all service nodes [86]. The proposed solution used directed acyclic graphs as a model for service flow graphs. The resulting federated service flow graph was resource efficient, and led to favorable performance results in terms of bandwidth and service latency. However, the solution did not consider service provider mobility thus rendering the solution impractical in highly dynamic mobile networks.

An ad hoc composition algorithm for service overlay networks was presented by Kalasapur et al. [87]. A graph-theory-based service composition mechanism was used in a hierarchical service overlay. The service model captured features and useful characteristics of resources in pervasive environments using PICO middleware. However, the system relied on a centralized semantic directory to register available services, be queried by incoming users, and perform lookup and composition on registered services. Reliance on a centralized solution introduces complications related to central point of failure, scalability, and bottleneck issues.

Our composition mechanism overcomes many of the limitations seen in previous works. We provide a decentralized service composition method that relies only on a MS-generated plan. Each intermediate MP searches for and forms its next-hop link until the MC is reached. A path that meets the MC QoS requirements is guaranteed. Additionally our solution limits the amount of broadcasted messages and restricts searches to known regions of the overlay. Furthermore, we provide a second plan-free composition mechanism. The composition solution utilizes semantic and syntactic nearness evaluations of service operations through which MC requirements can be met through semantic progression in the composition path.

3.4. Summary

This chapter discussed various approaches and addressed some of the issues associated with service composition and mobility. Reliance on a centralized composition scheme can provide reliable and fast discovery methods for services required in a composition. However, the dynamic nature of mobile networks presents issues regarding the amount of messages exchanged between arriving/departing nodes and the centralized registry. Moreover, the dependence on a centralized registry presents the possibility of bottlenecks and single point of failure. Consequently, any composition solution taken into consideration must follow a decentralized approach.

Additionally, the chapter illustrated the limitations faced in probing-based service discovery methods. Flooding query messages into the network has two major disadvantages. First, it depletes the bandwidth needed to deliver media streams and services. Second, it interrupts service nodes as they must evaluate whether received queries can be served by them. Consequently, a hop-by-hop service discovery method is more economical and accurate as a discovery solution.

Finally, it was illustrated that determining the feasibility of a service in an SSON composition, requires evaluating the syntax and semantics of discovered services. The evaluation is used to determine the similarity level between discovered services and those actually needed in the composition.

Chapter 4

Autonomous SSON Multi-Layered Architecture

This chapter gives an overview of our multi-layered system architecture designed to aid in simplifying the service composition process. Section 4.1 presents a general overview that discusses some of the challenges and limitations of existing systems. The three layers of the system are discussed briefly in Section 4.2. The architectural design of mobile nodes forming the presented system is presented in detail in Section 4.3. Section 4.4 gives a detailed presentation of our hybrid SON design and its associated algorithms. In Section 4.5 we present simulation test and their respective results. Finally, in Section 4.6 the chapter is concluded with a brief summary.

4.1. Overview

Our multilayer SSON composition architecture stems from the SMART project [14] discussed earlier. SMART was proposed for guiding media flow through specialized network nodes to utilize their ability to cache, transcode, synchronize multimedia data or any other service through the use of overlay networks in an Overlay Control Space (OCS). OCS is responsible for selecting the necessary media processing nodes and establishing a service-specific end-to-end overlay network for media delivery. Our architecture extends this design to form a multi-layered architecture constituted of the physical underlay, the SON overlay, the SON management overlay, and SSON overlay.

Overlay networks constitute a virtual network topology on top of the physical network. This applies to first-level overlay networks. However, multi-layered overlays can be constructed over existing overlay networks. There are numerous advantages to the use of overlay networks. First is their allowance for network developers and application users to design and implement their own communication protocols on top of the Internet. Second, data routing in overlay networks can be flexible, with an ability to detect and avoid network congestions by dynamically selecting paths according to different metrics. Third, unlike the physical network, overlay networks are characterized with scalability and robustness. Overlay nodes are highly connected to each other. In other works, even with node failures, most if not all nodes can remain reachable. This is a direct result of the flexibility of routing. Consequently, end-to-end communication in overlays can always be achieved as longs as the physical network is connected. Finally, large amounts of information and resources can be effectively shared due to the high connectivity and large number of overlay nodes.

However, overlay networks face several challenges. Due to the abstractive nature of overlays, overlay networks do not have the ability to control the physical network, nor do they have access to some of the critical information of the physical network. Additionally, overlay applications may result in inefficient usage of network resources. This is due to mismatches between the overlay and underlay topologies, and inaccurate probing results. The latter is reflected with a spike in the amount of messages exchanged. A third challenge is the vulnerability of overlays to security and privacy issues, a topic beyond our research work. Moreover, the decentralization of overlays weakens their ability for resource coordination. Finally, there is a noticeable weakness in research work around the issues of fairness of resource sharing and collaboration among overlay nodes. Requests for similar services should be distributed amongst MPs capable of providing such services. Limiting service request to a small number of MPs present deplete resources of heavy-working nodes while underutilizing resources of idle MPs.

Consequently, any overlay-based system architecture must embrace both advantages and disadvantages of overlay networks. The challenges facing any architecture can be summarized as follows:

1. Providing overlay management given the dynamicity of mobile networks. The dynamic changes in network conditions and topology, renders past management information obsolete. Changes occur on two interconnected fronts; the unexpected failure of nodes, and variations in QoS. The former reflects the limited topology knowledge possessed by each overlay node. Arrivals and departures of nodes require links' reconfigurations. However the restricted topology knowledge possessed by overlay nodes makes managing an overlay especially difficult as the number of overlay nodes and overlays increase. The latter; usually affected by congestion of routing paths, results in loss of bandwidth and increase in latency thus requiring link reconfigurations to redistribute the load on links.

   Any management scheme must account for the different phases of overlays: creation, optimization, adaptation and termination. We, therefore, refer the reader to the policy-based management scheme in [88] [89] [90] which we utilize as the basis onto which our architecture is built. A set of policies adapted to the current availability of resources and users' demands are dynamically generated from the available context information and enforced on the fly. These policies control the various construction phases to automate overlay management in a dynamic manner. In this chapter, however, we focus on the second challenge facing overlay management; the overlay's topology.

2. An important factor affecting the performance of overlay networks is their topology. A lot of research has been done regarding overlay topology issues. Given that overlay nodes are connected via overlay links (IP-layer paths), an overlay link theoretically connects each pair of overlay nodes. The variation in overlay links selections affects the overlay service quality and cost. Consequently, the topology chosen affects the overall delay experienced in searching for potential SSON MPs as well as the total number of messages exchanged. Therefore, the chosen topology should show a high level of resiliency for path failures achieving short failover times. Additionally the topology should improve routing performance when compared to other available topologies. It must reduce the required number of query messages. Most importantly it must be dynamic and adapt to the instabilities of mobile networks.

4.2. Layered Architecture Overview

The SSON construction multi-layered architecture is a direct extension of the SMART model for overlay networks from the Ambient Networks architecture [26]. An Ambient Control Space (ACS) was defined, acting as a common control layer to all resources and technologies in heterogeneous networks and facilitating media delivery. SMART (Smart Multimedia Routing and Transport Architecture) [14], Figure 2.1, was proposed for guiding media flows through specialized network nodes to make use of their ability to cache, transcode, synchronize multimedia data or any other service through the use of overlay networks in an Overlay Control Space (OCS). OCS is responsible for selecting the necessary media processing nodes and establishing a service-specific end-to-end overlay network for media delivery. It manages the creation of SSONs upon requests from users. It also controls the reorganization and adaptation of existing SSONs based on changes in the underlying ANs.

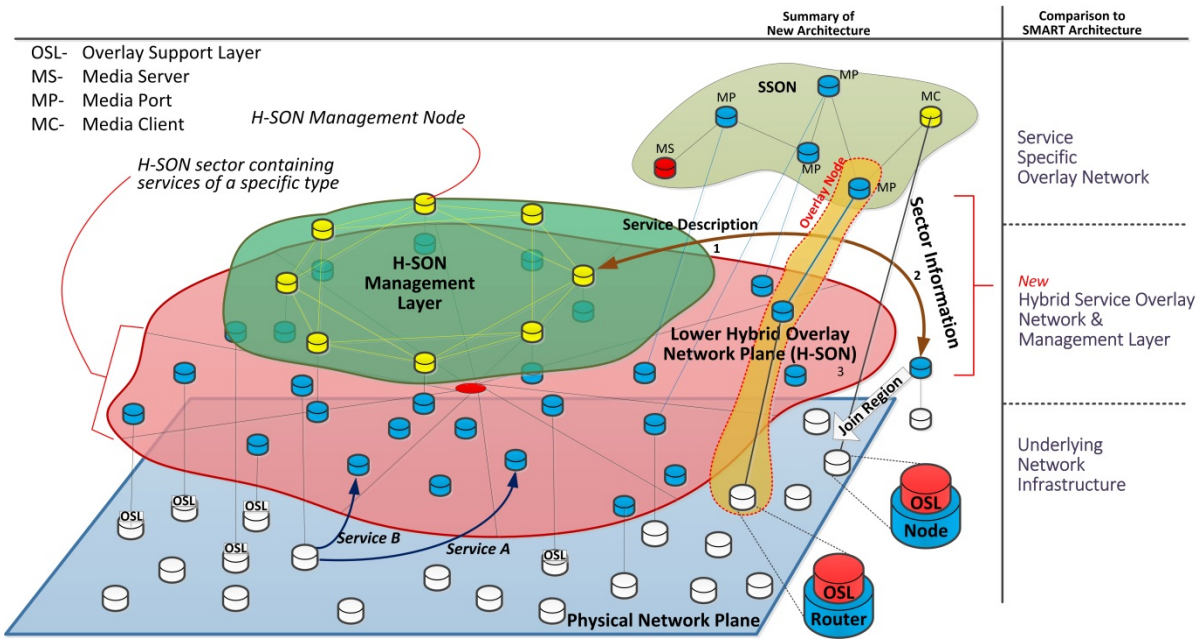


Figure 4.1 Three-layered SSON composition architecture.

The Overlay Support Layer (OSL) provides the functionality needed for the establishment of overlay links between SMART nodes. It is responsible for handling

packets on the overlay; thus sending, receiving and forwarding packets on the overlay layer. This is controlled by SSON-specific routing tables. In this chapter we discuss our 3-layered SSON composition architecture illustrated in Figure 4.1. The three layers are: the physical underlay, the hybrid SON layer, and the SSON layer. The majority of the discussion is dedicated to our unique hierarchical hybrid SON that facilitates SSON construction in the layer above with minimized delay and number of exchanged query messages.

4.2.1. Physical Underlay Level

Typically, overlays build a virtual topology over specific network technologies. The components of a network technology are usually layer 3 (network layer in OSI model) boxes, such as IP routers. However, we loosen the definition of network components to include any devices capable of providing a service and that are part of the network. The underlying network should provide the necessary connectivity and QoS for the media flowing between network components. This is typically done using QoS technologies such as DiffServ [30] or IntServ [31], multi-path routing [91] [92] [93], or a variety of traffic engineering methods.

Although one of the main goals of the SMART project in AN is to provide a higher level of abstraction when constructing each SSON flow. However, the following must be taken into consideration when designing an SSON composition system:

1. All network components are assumed to be mobile. Consequently, the possibility of a node entering or leaving the network unexpectedly is high. Additionally, nodes can move in different directions, loose contact with some neighbors or add new contacts. The above SON virtual layer must handle the dynamicity of the underlay.
2. Network devices are designed and managed by different vendors. Thus, internal protocols and resource capabilities differ from one component to another. This heterogeneous environment must be abstracted by the above virtual layers to provide a uniform view of the network. This results in a generic and extensible framework for service composition.

4.2.2. Hybrid SON Level

A virtual network layer is built on top of the physical underlay to abstract unnecessary details from the SSON composition process. This separation layer allows for a cleaner distribution of the problem space. At the bottom is the underlying physical network that handles the specifics of connectivity and QoS. On the other hand, at the top is the overlay SSON layer with overlay (virtual) nodes that contain the necessary state to associate individual flows to a specific SSON. The overlay level adds a certain level of media-awareness onto which an SSON can be constructed to instruct Media Flows to be routed and undergo processing in order to meet service requirements of users.

Our Hybrid Service Overlay Network is discussed in detail in Section 4.4 of the dissertation. The main goal of the H-SON is to reduce the experienced delay, the number of distributed query messages, and to increase the accuracy of MP searches in SSON composition. We rely on a hierarchical topology consisting of a central Chordal ring from which radiate multilayers of Chordal rings and unstructured node connections. The reliance on the hybrid topology stems from its stability, fast healing capabilities, and QoS-based node positioning for fast MP searches.

4.2.3. SSON Level

The Service Specific Overlay Network level is defined by a set of overlay nodes and overlay links constituting a virtual network path. These topologies are dynamically configured on top of the H-SON topology. SSONs are customizable and possess the flexibility to be constructed according to specific requirements of media delivery services to a MC. SSONs are constituted of network-side nodes; MPs, that allow for modifications of the content and services. Thus, SSONs make it possible to support media routing, adaptation, distribution and caching.

The construction of SSONs requires modeling the MC media description and QoS requirements, discovering potential MPs that meet these requirements, and forming a composition path that delivers the required media. Section 4.4 of this chapter illustrates a use case scenario that utilizes our H-SON to construct a possible SSON. We illustrate how a

composition path is formed by searching for MPs according to H-SON sectorial divisions, and focus the search area to MPs that meet MC QoS requirements by restricting queries to specified bands within the valid sectors. Given that discovery of candidate MPs requires semantic and syntactic knowledge of component services' functionalities, a clear modeling method of the service functionalities, inputs, and outputs is required. Furthermore, differentiating potential services and choosing those that are nearest to the SSON's needs is a difficult problem that must be solved. Consequently, the results of this chapter are expanded upon in the next two chapters to provide a complete solution for planning an SSON path, discover potential MPs, choose the most-fit MPs, compose service-specific paths, and deliver media streams to MCs.

4.3. Architecture for Autonomic Management of Overlay Networks

The emergence of the service centric paradigm of the current Internet due to the advent of plethora of new services, multimedia services in particular, has raised questions over the traditional way of treating networks as "dumb pipes". Charging the customers a flat fee for using network resources does not guarantee the return on investment for network operators which hinders network upgrades and expansions, in turn affecting the QoS provided to different services. On the other hand, the "walled garden" networks restrict the emergence of new services as well as their flexibility due to network operators' lack of expertise in the nuts and bolts of services and the service providers not disclosing crucial information about their services. The solution is to separate network infrastructure management and service delivery so that they can be managed independently. In parallel to service delivery, there has been substantial growth in the network hardware as well, which has led to the increase in heterogeneity of network infrastructure, e.g., Optical fiber and Wireless. Therefore, any service is likely to go through a variety of transmission media for an End-to-End (E2E) service delivery. Increased heterogeneity has challenged the service providers in meeting customers' QoE expectations. The above scenario has also meant that network failures are common events in modern networks. Overlay networks were introduced to address the above issues. Therefore, we utilize overlay networks to separate the management of the underlying

network and the services provided. In this Section, we present a new architecture and methodologies to manage such an overlay network for media delivery, where service providers pay to use network resources.

The essence of Autonomic Computing (AC) [94] is incorporated in the presented architecture to achieve certain degree of self-management with minimal user intervention. AC refers to the self-managing characteristics of distributed computing resources, adapting to unpredictable changes whilst hiding intrinsic complexity to operators and users. The ultimate aim of AC is to develop computer systems capable of self-management. Self-management refers to self-configuration, self-optimization, self-healing and self-protection. A system that possesses AC capabilities is called an Autonomic System (AS). An AS is driven by business goals called high level policies whereby, it can make decisions on its own, optimize its status and optimally adapt to changing conditions. An AS is generally modeled with two main control loops (global and local) with self-monitoring capabilities aided by sensors and self-adjustment capabilities aided by effectors. An Autonomic Control Loop (ACL) consists of monitor, analyze plan and execute activities.

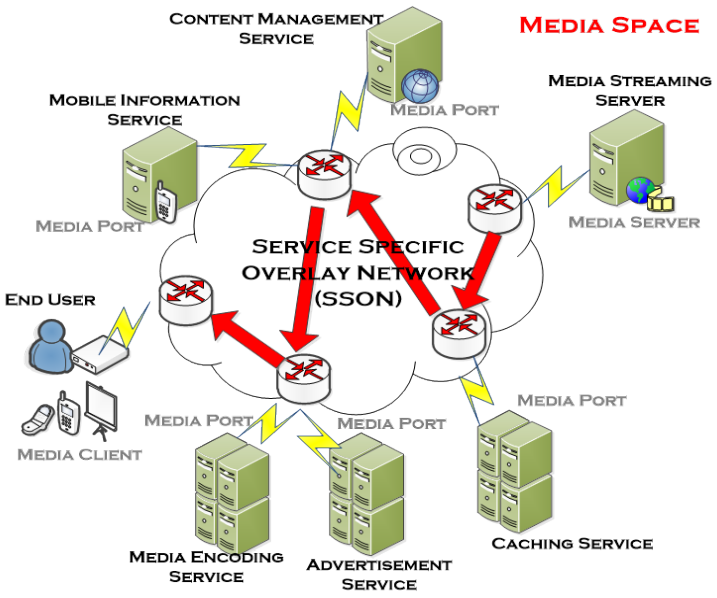


Figure 4.2 Network scenario of the SSON composition architecture.

We have presented the architecture illustrated in Figure 4.1 in some of our published works [95] [96] [97]. Figure 4.2 depicts the network scenario where the architecture is applicable.

It illustrates how an application originating from a Media Server (MS) is provided as a composition of a number of services provided by different Media Ports (MPs) to finally deliver the service to a Media Client (MC). It defines how an SSON is formed by optimally selecting the nodes to represent MPs. In this approach, an SSON is built for each user when a request arrives; hence maximum QoS guarantee is provided for each request. SSON construction uses network side functions called MPs that have the flexibility to modify content and services such as caching, adaptation and synchronization. SSONs are built to provide composite services based on technical, QoS, and QoE requirements of MCs for basic services originating from MSs. The architecture has provisions to further improve node selection and service compositions besides providing means for autonomic management of the whole overlay which includes the complete ACL.

4.3.1. Architecture Design

The use of overlays facilitates the isolation of services management from the underlying hardware complexities. It also helps seamless management of heterogeneous networks. In order to meet the design goal of scalability, a semi-decentralized management approach is embraced in our architecture, which also facilitates having a global and a local ACL. Figure 4.3 illustrates the autonomic overlay architecture as published in [95]. As it can be seen, the scope of this section is restricted to the management of overlay and SSON only. A SSON forms a media space which consists of a MS, a number of MPs and a MC. Each MP is an autonomic component that consists of an Autonomic Manager (AM). Therefore, each MP's functionalities complete a local ACL. The Autonomic Controller (ACon) is the central entity in an SSON that provides device level policies to each MP to control their functionalities. The sensors in each MP provide the sensory data to ACon's Network Monitor (NM). The NM uses this data to monitor and update the current state of the network.

The overlay network is composed of both structured and unstructured nodes. In structured overlays, nodes are organized following specific criteria and algorithms, which lead to overlays with specific topologies and properties. Unstructured peer-to-peer networks do not provide any algorithm for organization or optimization of network connections. A hybrid structure forming our lower overlay network is used to overcome limitations seen in

both structured and unstructured overlay networks and to maximize their advantages. Our overlay structure consists of several hierarchical layers of Chord-like rings and unstructured leaf connections. Details of the hybrid SON are discussed in Section 4.4.

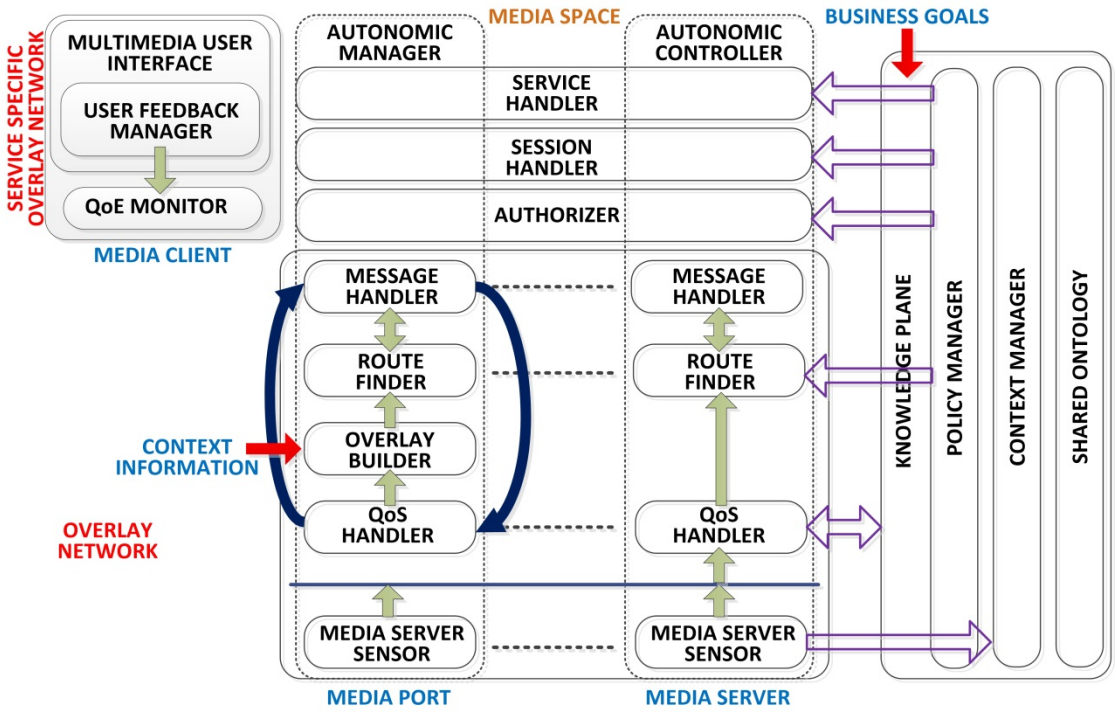


Figure 4.3 Autonomic overlay architecture, as published in [95].

4.3.1.1. MediaPorts

A MP offers a particular type of service to users like caching, synchronization video encoding/decoding and content insertion. Our work does not assume a specific Service description. Services can be described using standard Web Service Description Language (WSDL) [98] for example, and extended with semantic meta-data.

MPs can be described according to their input and output ports: single, splitters or joiners [99]. Single MPs have only one input port and one output port. They take a media flow as input and transform it into a different output flow according to the service function they offer. Splitters have one input and several outputs. They take one media flow as an input and produce a number of output flows. Joiners have several inputs that they merge into one

output. Services can therefore be independent, partially or completely composed. Independent MPs can perform a service without help from other MPs. Partially composed MPs consist of at least two MPs where the outputs and inputs of the first can be composed with some of the inputs and outputs of the second. They are partially composed in the sense they need other MPs to provide a complete service. Completely composed MPs are made up of at least two MPs where all output and input ports of the first are composed with all input or output ports of the second. Completely composed MPs are also those where all outputs of the first are composed with some inputs of the second and where the remaining inputs are composed with all inputs of a third MP. In other words, completely composed MPs are those that provide a complete service.

4.3.1.2. Autonomic Manager

Each overlay node is a potential MP. Each one of them is equipped with an AM. As illustrated in Figure 4.3, an AM handles both the overlay's and the SSON's functionalities. An AM consists of Media Port Sensor (MPS), QoS Handler (QH), Overlay Builder (OB), Route Finder (RF), Message Handler (MH), Authorizer, Session Handler and Service Handler modules.

A Media port sensor consists of a Resource Manager (RM), a Performance Manager (PM) and an Energy Manager (EM). A RM senses the usage of resources within a MP such as storage space, processing power and main memory. RM is used to obtain the availability of resources in a MP. The PM keeps track of the performance levels of services provided by an MP. An EM on the other hand evaluates energy amount consumed by a MP based on resource usage data and an Energy Profiler (EP). The EP is a table that gives the energy consumption for different levels of resource usages. The sensory information is stored in a Context Manager (CM).

QH's responsibility is to evaluate the QoS (service delay) levels of services a MP provides based on the inputs from RM and PM. QoS of each MP is evaluated based on the score assigned to the respective overlay node, based on a number of criteria which includes node stability, load, energy efficiency, QoS levels and security. The calculation of the score

is explained in details in Section 4.4. A virtual Chord-like structure is maintained based on the scores given to overlay nodes. Nodes with high scores are in the center while lower-scored nodes are farther away from the center of the Chord structure. The SSON is built by optimally selecting MPs that form the path from the MS to the MC and also facilitate service composition. Stability of a node is given by the "life time" of the respective node. We assume that the longer a node remains in the system, the higher is its stability level. The energy consumed can be minimized if idle MPs can be put to sleep. To achieve that, we tend to avoid using an unused MP. Although it may be expected that the longer a node remains within the network the more likely failure occur due to power depletion. However, we make the assumption that nodes have a large reserve of power especially given recent advances in battery design. Future extensions of our work will remove this assumption and deal with resource-limited nodes to solve the service composition difficulties associated with such devices. Security of a node is measured using a trust based evaluation [100], where a trust score is given to each node based on the extent other nodes trust the respective node. This is inspired from the trust in human relationships. The OB module is responsible for positioning MPs in the overlay and hence building the overlay network in a distributed manner.

The formation of the hybrid overlay network is performed in three stages: initialization, sub-Chord formation and extension, and promotion-based stabilization and overlay management. Initialization occurs when the first '*m*' nodes start arriving forming a central Chord structure. Nodes continue to join Chord until C_{Thresh} threshold is reached; a network dependent variable related to the expected number of nodes in the network, available bandwidth and other criteria. A *MP Initializer* triggers the MH to send a join request message to a nearby node chosen randomly and await a response message that includes the node's successors and finger links. If a response is received then the OB instructs the MP to join the Chord Structure. Otherwise, the process is repeated and a join message is sent to another Chord node until an acceptance is received. Once a C_{Thresh} number of nodes have joined, the central Chord network is formed and the *MP Initializer* in an arriving node must request a join from one of the existing Chord nodes. Receiving an acceptance, the node is added as a minor node to the responding node. Some of the crucial tasks of OB are as follows: 1)

joining the overlay network, 2) minor node reorganization, 3) node failure handling, and 4) media space management.

RF is responsible for finding the best possible route from a MS to a MC connecting a number of MPs to facilitate effective service composition. In the work presented in [32], search for MPs to form a composition was performed based on three assumptions: The location of the MC is known, the location of the MS is known, and the search for MPs is performed in a direction emanating from the MC towards the MS only. Location information is provided by the CM. Using this, any node's RF receiving a query will forward the query to local nodes only if it is within α degrees. The angle α represents the maximum search scope for possible component services. This value depends on the locations of the MC and the MS. This approach, in general, avoids sending queries to regions of the network where answers are not likely to be found. However, in wireless mobile networks, the dynamic movement of nodes at the network layer may result in an unequal distribution of MPs at the overlay layer. We may end up with high density regions where a large number of diverse MPs exist while other regions may have sparsely located services. In the latter case it is highly likely that α must be continuously increased until all component services are found.

In our work, we overcome limitations of the previous solution using two methods. The first is by providing a hybrid lower overlay structure discussed earlier. The second is by dynamic node promotion and demotion according to node scores. These two methods result in a circular MP space consisting of MPs and MSs that is further reorganized by the MP Organizer module residing in OB of MPs based on the types of provided services. As a result, MP Initializer modules of nodes arriving in the network must first determine the overlay region they must join. We assume the following: 1) the center of the MP space is a known region that can be located by all nodes joining the network by querying the CM, and 2) angles α and β representing the start and end borders, respectively, of each service type sector are also well known by RFs in all nodes joining the network. α is measured from a *0* degree angle pointing to a relative direction accepted by all nodes in the MP space.

The *Authorizer* module is responsible for authorizing other MPs to receive and provide services. This is performed based on the trust each respective node has earned from past negotiations. If the trust score is above a predetermined threshold, then communication with

the corresponding MP is authorized. Trust score is assigned based on history, i.e., the more a node has provided required services in the past, the higher is its score.

The *Session* handler module is responsible for establishing an overlay session to initiate an end user application process, maintaining it and terminating it. The Session handler's main task is to uphold the end-to-end QoS levels of applications. In cases where there is a drop in QoS levels, the Session Handler informs the Service Handler which does the necessary service specific configurations to keep QoS at desired levels. This module is also responsible for triggering RF, when there is a failure in the established path, which in turn finds an alternate path.

The *Service* Handler manages services the MSs and each MP provide by appropriately configuring them to meet users' requirements. This includes, for example, altering the resolution of provided videos.

4.3.1.3. Knowledge Plane

The Knowledge Plane (KP) is responsible for providing the AC with all required knowledge. The knowledge to form KP in a distributed manner is provided by context information, ontologies [101] [102] and a set of policies. An ontology provides a shared vocabulary, which can be used to model a domain, that is, the type of objects and/or concepts that exist, their properties and relations. KP infers or generates new knowledge from acquired knowledge, which then goes through a thorough analysis for conflict detection and removal to maintain consistency.

The responsibility of the CM is to provide network state information from monitored data. CM is equipped with the necessary intelligence to obtain service level status information from the device level status information. This is processed by gradual inference of statuses at different levels. At the very bottom level, CM acquires the status of network resources from the NM. It then gathers the status of services provided by MPs, e.g., QoS, and their inter-relationships. Finally, the composite service status is obtained from the status of individual services. Some examples of composite service are QoE and energy usage. The Policy

Manager (PolM) is responsible for setting and enforcing a set of policies required to manage the network system in a desirable manner.

4.3.1.4. Autonomic Controller

ACon is a top level autonomic component in the hierarchical network management entities, residing in MSs that control the functioning of networks by feeding the AM in MPs with instance level policies. ACon is fed with business goals by the network administrator. The high level goals constitute high level policies. AM refines these high level policies into instance level policies that define the MP's objectives. ACon contains all the elements of AM in addition to the NM module. Since each SSON has one ACon, it controls the management of SSONs in a centralized manner, while the management of each MP is performed by the respective AMs.

The NM performs state monitoring of the network by analyzing frequent input data it receives from media port sensors. The data it receives include QoS levels, resource consumption, energy usage profiles and failure occurrences. Furthermore, NM is responsible for monitoring QoE levels of individual services based on utility measures given by the QoE Monitor in MCs. NM is the centralized monitoring entity in an SSON; hence, its primary focus is the monitoring operations of multimedia applications.

4.3.1.5. Multimedia User Interface

A Multimedia User Interface (MUI) resides in MCs to facilitate users' interactions with applications in terms of selecting desired applications and content, and providing ratings on their experiences and content quality. Typically a MUI is equipped with the display screen (e.g., TV) and controller (e.g., infrared remote control) to control the application.

The User Feedback Manager (UFM) is a software that is placed with each MC to enable users to provide ratings on the content and applications they receive. Users are provided with rating options in pre-determined ranges (e.g., 1-10) on MUI, where users can cast their ratings via the controller. Users' ratings are taken into account to determine the popularity of content

and reconfigure application-specific settings to provide service levels that are as close as possible to users' expectations.

The QoE Monitor is responsible for monitoring the user's QoE. QoE is a subjective measure of a customer's experience; therefore, it is very challenging to get an accurate measure of it. QoE is represented using a sigmoid utility function that takes into account the QoS parameters (e.g., byte loss, delay and bandwidth) that can be measured at the MC. We make use of previous work in [103] [104] regarding a loss-based utility function to predict QoE. We refer the reader to [105] [106] [107] for more details on the QoE monitoring function and simulation results.

4.4. Hybrid Service Overlay Network Layer (H-SON)

Each overlay node in the SON layer can be one of three types: MediaClients (MC), MediaServers (MS), and MediaPorts (MP). MCs represent users' nodes requesting a media flow from MS source nodes. MPs provide network-side media processing capabilities and transformation services on the media path between the source (MS) and the sink (MC). These MPs transform the multimedia data from the MS into an MC-acceptable form. These transformations take into account the available context information for optimal service delivery. The responsibility of forming a service composition is performed at the SSON layer. However making SSON construction a speedy process with minimized numbers of exchanged messages requires an optimal topology organization of overlay nodes. As such, we present a hybrid SON (H-SON) connecting MSs, MPs, and MCs. In this overlay, nodes of similar services are clustered in well-known regions. Additionally, the nodes are organized such that those with higher QoS are located closer to the center of the MP space (hybrid overlay) while those of lower quality are dispersed further away from the center. This organization improves response time, reduces overhead and number of overlay hop counts required to provide all requested services. This is done in a way the network's resiliency to dynamic mobile topologies is improved. In such a network, overlay nodes, and their respective underlay mobile service nodes are constantly in motion with unpredicted arrivals and departures.

The justification for promoting use of a H-SON stems from the following reasons:

1. Searching for component services to form a composite service according to an expected level of QoS may require a large number of message exchanges between nodes until the required node is found.
2. When forming SSONs, it is beneficial to find MPs in the same direct path from the MS to the MC, provided required QoS for each service is attainable, and that discovered MPs have a low hop count. An SSON-level low hop count reflects a low number of hops in the network level as well, thus reducing delay.

4.4.1. Standard and Optimal-Chord Rings

Prior to discussing the details of our H-SON, we note our use of optimal Chordal rings in the structured sections of the topology rather than standard Chordal rings. Given the importance of providing the most simple and expandable Chord rings in the structured section of our H-SON, we have opted to utilize a widely-studied family of degree 4 Chordal rings. They are also called optimal Chordal rings due to their symmetric network properties, high fault-tolerance, low broadcast time and ease of routing [108] [109].

Each node in an optimal Chordal ring is connected to 4 neighbors; two ring nodes and two Chord (finger) nodes. The ring nodes refer to immediate predecessor (i-1), and successor nodes (i+1) on the same ring. The overhead introduced by the two finger nodes compared to unidirectional ring structures is offset by the benefits of a low diameter. Diameter here refers to the longest direction possible from any two nodes members of the Chordal ring. Additionally, finger arcs simplify the routing process and increase the structure's resiliency. Optimal Chordal rings were first defined by Narayanan et al. in [109]. The formal definition is given as follows:

Let N and C be two positive integers, where $2 \leq C \leq N$. The graph $R_k(N, C)$ is an optimal Chordal ring of degree 4 with N nodes such that the node set is $\{0,1, \dots, N-1\}$ and the edge set $\{[i, (i+1)\mathbf{mod}N], [i, (i+c)\mathbf{mod}N], i \in \{0,1, \dots, N-1\}\}$. This is the case

$$if \begin{cases} N = 2k^2 + 2k + 1 \\ C = 2k + 1 \end{cases} \text{ AND} \tag{4.1}$$

where *k>1* is the ring's diameter. The symmetry of optimal Chordal rings makes all nodes equivalent. Figure 4.4 illustrates an optimal Chordal ring $R_2(17, 5)$; i.e. a Chordal ring with 13 nodes, each having 2 finger links, and the distance between the node and its first finger is 5 nodes.

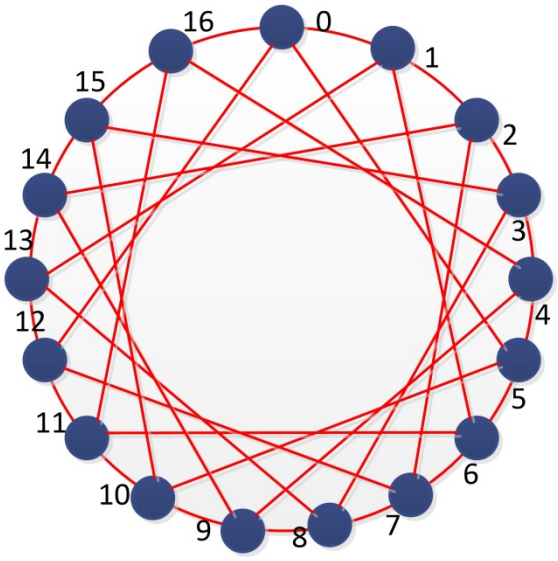


Figure 4.4 Optimal Chord ring $R_2(17, 5)$.

4.4.2. Hybrid Hierarchical Chordal Rings and Node Classification

Our hybrid structure forming the SON is used to overcome limitations seen in both structured and unstructured overlay networks. Unstructured overlay networks are formed with arbitrary overlay links. These networks are easily constructed. However, there is a notable limitation when a peer is interested in acquiring specific data. In this case, the query needs to be flooded in the network to find a node holding the desired data. The less popular the service a client is interested in acquiring, the less likely it will be found in a nearby node to the requestor. Consequently, the number of messages exchanged would consume significant bandwidth in a wireless network in addition to the operational overhead. Moreover, since there is no correlation between the service requested and the corresponding MP, it is highly unlikely that a search turns successful.

Structured overlay networks overcome many of the limitations of unstructured networks by maintaining DHTs and distributing the content in these networks. The disadvantage of structured overlays is the large number of messages that need to be exchanged to stabilize the network, maintain existing links, and heal links to failing nodes that drop from the network unexpectedly. This rigidity is highly unfavorable in dynamic mobile environments where nodes leave the network without warning and new nodes are arriving constantly.

Hence, we suggest the use of a hybrid overlay structure that maximizes the advantages of both structured and unstructured overlays and minimizes their disadvantages. Our overlay structure consists of several hierarchical layers of Chord-like rings and unstructured leaf connections. Each node in the network can be one of two types; a *major node* or a *minor node*. A Major node is any node that has been deemed capable of joining one of the Chord structures present within the network. These nodes should have the following characteristics:

- Have a long life span and are highly likely to remain present in the network. This is directly related to the node's arrival time. The dependency of other nodes' links to a major node indicates the possibility of negatively affecting the overlay's performance in the event major nodes fail or unexpectedly leave the network. Consequently, failures within Chordal rings must be kept at a minimum. This is achieved by promoting more stable nodes to be members of the various Chordal rings of the H-SON.
- Have an adequate level of available resources to support links to nodes present within the same Chord structure in which they are located, be the managing nodes for further sub-Chord structures and be the parents for other leaf nodes (nodes only attached to a single parent node). This requires large amounts of resources such as fast processors and memory, as well as capability to serve multiple links simultaneously.
- Have popular services frequently requested during service compositions. The more popular a service becomes, the more crucial it becomes for it to be highly contactable by potential overlay nodes for SSON compositions. This is assured by a high node degree. Consequently, having multiple direct neighbors, which is the case for a non-leaf node, indicates the possibility of forming composition paths that avoid bottlenecks.

Major nodes are divided into two types: *Normal Major Nodes (NM)* and *Link Major Nodes (LM)*. An NM is a major node that may have any number of child leaf nodes, child Chord nodes, and neighboring nodes. NM nodes do not link directly to any higher level Chord node. LM nodes on the other hand are major nodes that have the same characteristics as NM nodes with the addition of a link to a higher level Chord node. As a result, LM nodes act as link nodes between a higher level and a lower level Chord structure. A Minor node, or a leaf node, is any node that is linked to only a single node acting as its parent. Minor nodes share the following characteristics:

- They are newly arrived nodes that have not built stability credentials in the system. As such, they are assumed to be unstable nodes whose services may be short-lived. Due to the high probability of unwarned departures or failures of unstable nodes, placing these nodes as leafs minimizes their ramifications on the overall performance of the H-SON. No link dependencies exist on leaf nodes, and the only overlay node that must detect a leaf's departure is its parent.
- They are nodes that do not have the necessary resources to support links to other neighboring nodes if they were part of a Chordal ring. Forwarding queries and media packets to neighboring nodes requires a certain minimum amount of resources. As a result, nodes that lack the necessary level of resources must remain as leaf nodes.

  They are nodes that have unpopular services rarely requested in compositions. Consequently, these nodes do not require a high connectivity degree. The lack of interest in such services renders the need for multiple connection paths to leaf nodes wasteful.

A generalized H-SON is illustrated in Figure 4.5. The topology consists of a central Chordal ring encompassing the most stable nodes, offering the highest level of quality for their services, and having the greatest amount of available resources to serve composition requests. Each node in this Chord structure can be either an independent node or an ancestor node. In the former case a node is only linked to its neighbors within the same Chord ring. The node does not have any child leaf nodes whose connections depend on this Chord node. In the latter case however, the Chord node has one or more child (dependent) nodes connected to it. Connected child nodes can be either single leaf nodes, or LM nodes that bridge to a lower Chord ring. From this H-SON center, a series of hierarchical layers of

Chord rings radiate towards the outer edges of the network. MPs with decreasing amounts of available resources and QoS levels are found further away from the center.

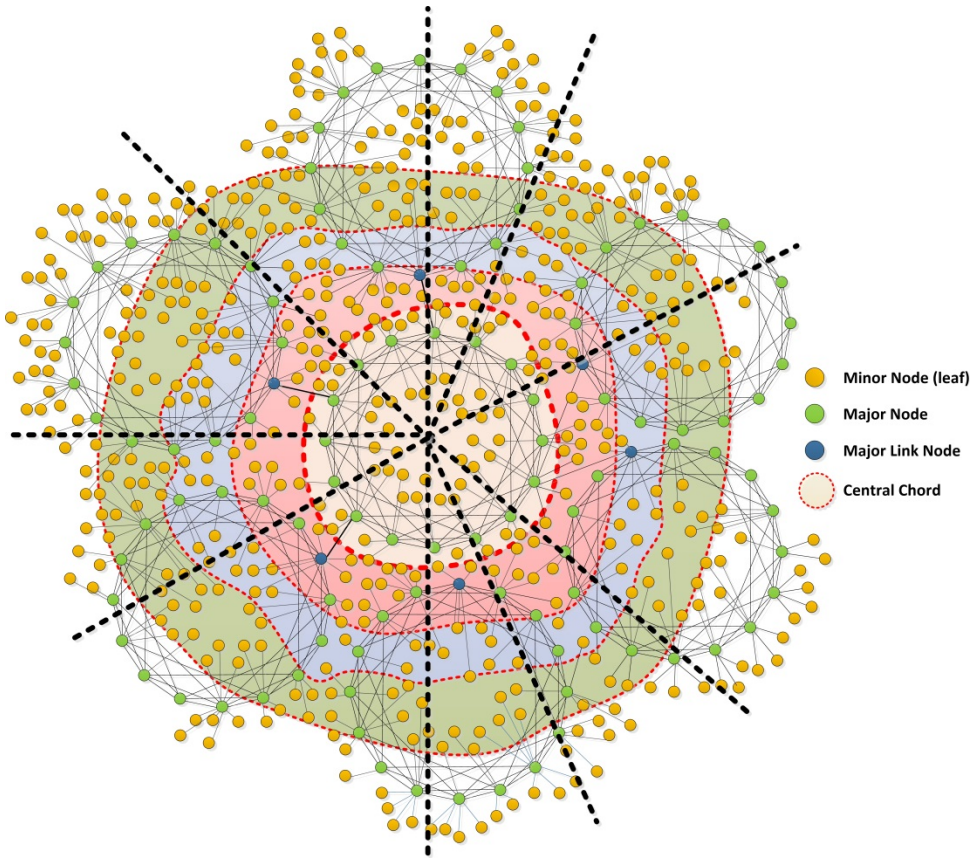


Figure 4.5 Hybrid service overlay network (H-SON).

4.4.3. Constructing Chordal Rings in Hybrid SON

The formation of the H-SON is performed in three stages: Initialization, Sub-Chord Formation Extension, and Promotion-Based Stabilization for overlay management. The following subsections discuss details of the respective phases.

4.4.3.1. Initialization Phase

The initialization state occurs when the first '*m*' nodes begin arriving into the network thus

forming a central Chord structure. Nodes continue to join the Chord ring until C_{Thresh} threshold is reached. This is a network dependent variable established by the H-SON management layer. The value of C_{Thresh} represents the maximum number of neighboring overlay nodes that can link together to form a single Chord ring. The C_{Thresh} value is chosen to meet the following goals:

- Reduce the number of interconnected nodes affected by unexpected failures of neighboring nodes.
- Reduce the number of stabilization messages exchanged to maintain an updated view of current nodes in the Chord ring.
- Reduce the number of hops required to transfer a query from a higher priority Chord ring to a lower one, or vice versa.
- Maintain a balance between the structured and unstructured characteristics of the H-SON. With increased structuring, the overlay becomes less dynamic and less adaptable to the instabilities of mobile environments. Conversely, with decreased structuring, a significant increase in broadcasted query messages is noticed for successful MP searches.

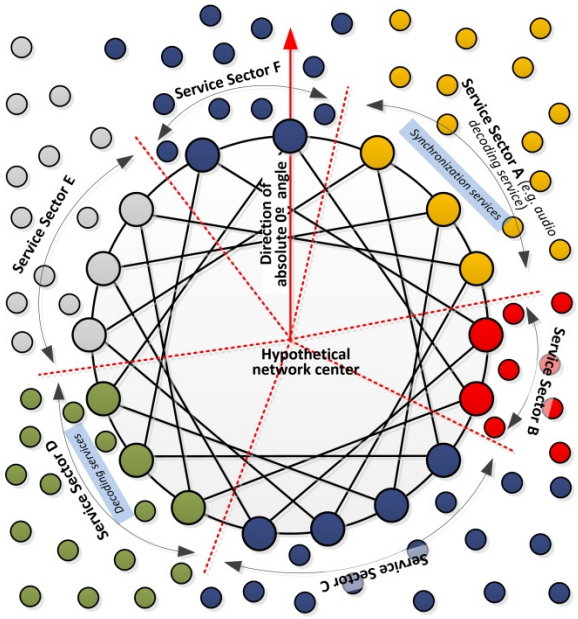


Figure 4.6 Service type-based sectorial division of the hybrid service overlay network.

All nodes belonging to the same Chord structure have an equivalent type of service being offered. Moreover, they provide closely matching levels of QoS. Most importantly, nodes within the same Chord ring have closely matching stability levels. Consequently the probability of an unexpected failure of each node in the ring is somewhat equivalent to its neighbors.

The above process leads to the formation of a central optimal Chordal ring. The location of each node however must follow a service type-dependent method of clustering. As illustrated in Figure 4.6, synchronization services (*yellow*) are located in one section of the ring, while media decoding services (*green*) are located in another, and so on.

4.4.3.2. Joining the H-SON and Score-based Sub-Chord formation

Every arriving node sends a join request message to a nearby node chosen randomly and awaits a response message. A sample join request message is given as follows:

$join_{request}(MP_{id}, S_{MP}^{n}, QoS_{MP}^{n}, TTL_{response})$, where MP_{id} is a unique identifier the MP possesses. The identifier is only used for the initial join stages between the arriving MP and any nodes it contacts while the join process is ongoing. Each arriving MP can provide one or more services. The MP must provide a full description for all provided services $S_{MP}^{n} = \{S_0, S_1, \dots, S_n\}$. Similarly, each of the *n* services provided by an MP has its unique levels of QoS; $QoS_{MP}^{n} = \{QoS_0, QoS_1, \dots, QoS_n\}$. The details associated with modeling and describing services and QoS are discussed in Chapter 5 of this dissertation. The join request also includes a Time-To-Live timer ($TTL_{response}$) representing the maximum time an arriving MP is prepared to wait to receive its join response. An expired timer signals the loss of the join request message. In case of a message loss, the MP resends the message to the same initially contacted MP or randomly chooses another nearby MP. If an acceptance response is received then the node joins the network. Joining a Chordal ring follows the exact same algorithm used in normal Chordal structures. Section 4.4.6 discusses all algorithms used in our H-SON.

Note that not all nodes arriving into the network prior to reaching C_{Thresh} are added as members of the central Chord structure. As mentioned earlier, H-SON is divided into radial

sectors for each unique generalized type of service. Furthermore, each of these sectors is divided into QoS and stability-based intra-sector bands. Consequently, nodes arriving during the initialization phase may not possess the requirements to join the central Chord. As such, each of these nodes latches itself as a leaf to one of the central Chord nodes within the appropriate service-based sector band. This process results in the addition of a second level of MPs belonging to a sector band with less restrictive criteria for QoS and stability. Adding and managing the various levels of our hierarchical H-SON are discussed next.

To maintain the stability of Chordal rings in the H-SON, nodes periodically exchange messages to monitor the departure of neighboring nodes and to fix any broken links. However, due to the dynamicity of mobile networks, the negative side effects of mobility are usually reflected on the stability of overlay topologies. Consequently, we have added a new metric each node must exchange with its immediate overlay neighbors. The metric reflects node's stability and the supported QoS for provided services. The score for each node can be evaluated as follows:

$$NodeScore = \left[w_{arriv}(T_{curr} - T_{arriv}) + w_{serv} \sum_{i=1}^{n} P_i \left(\sum_{i=1}^{m} w_{Qual}{}_j Q_j\right)\right] \quad (4.2)$$

where, w_{serv} is the weight given to the importance of services provided in comparison to node stability for positioning MP in the H-SON. Serw_{arriv} is the weight given to the node's total time it has been present within the system since its last departure (if any). Time is measured using a global clock. The longer a node remains present in the system, the higher its stability credentials become. As such, a MP is recognized as a stable node if it is given a higher score value. In real-life situations, the probability of node failure increases the longer it remains within the network due to battery charge depletion. Consequently, it is expected that the battery health of MPs connected to the H-SON will follow the "Richard's Curve", a generalized logistic function [110]. It is a sigmoid function for decay extended from the widely used logistic curve and whose function is presented in (4.3).

$$Y(t) = A - \frac{K-A}{\left(1+Qe^{B(t-M)}\right)^{1/v}} \quad (4.3)$$

where *Y* represents the battery charge level at time *t*, *A* is the lower asymptote or minimum battery charge within the device before failure, *K* is the upper asymptote representing the battery charge level when the MP first joins the H-SON, and *B* is the battery charge decay

rate. Charge decay is dependent on the battery type, hardware used, and the number and type of services currently running on the device. $v > 0$ controls the asymptote point at which maximum decay occurs, Q is a variable dependent on the value $Y(0)$, and M is the time of maximum decay if $Q = v$.

The node score equation of (4.2) can thus be enhanced with (4.3) by replacing stability with a evaluating the total expected time *t* it will take an MP to deplete its charge, $Y(t) = 0$. Therefore the most stable node is the one with the highest initial charge level *K*, lowest failure charge level *A*, and lowest discharge rate *B*. For simplicity we maintain our use of (4.2) and reserve extending it through sigmoidal functions for future work.

The H-SON management layer (See Figure 4.1) can additionally act as a global clock that synchronizes the behavior of MPs in the H-SON. The creation of the management overlay precedes the presence of the H-SON, and some direct network administration intervention is involved in synchronizing the management layer MPs. All subsequent nodes arriving to join the H-SON start by contacting the H-SON management layer to determine the join sector and band areas where it must join and in the process the MPs synchronize their clocks with the management layer. Subsequently, when an MP first joins as a Minor (or as a Major node), then its parent (or neighboring node) begins monitoring how long the node remains present in the network beginning with T_{arriv} timestamp first recorded by the management layer and sent by the arriving MP in its *JoinRequest* message. Periodically, the management layer updates the sectorial and node score boundaries. During the exchange of update messages with the H-SON nodes, the MPs' clocks are synchronized.

Each MP can provide one or more services; up to *n* services, each with a specified popularity level P_i. Popularity reflects the frequency with which a service is used in SSON compositions to serve MCs' requests. Additionally, each service can have a number of QoS criteria Q_j. Quality criteria is dependent on the type of service being provided. For example, an audio encoding converter can have QoS criteria reflecting the total delay and the accuracy of its conversion process. Each QoS criteria is given a weight reflecting its importance. Consequently, if delay is the most important QoS property for an encoding service then its w_{Qual_j} value is larger.

Each Chord ring within a band in the hierarchical overlay has a minimum threshold,

$minThresh_i^{band}$ where i is the Chordal ring identifier. When a node's score falls below threshold, it is demoted to a lower Chord ring or becomes a minor node to one of its neighboring Chord nodes. Similarly, if a minor node's score in any Chord ring exceeds a maximum threshold $maxThresh_i^{band}$ set for that ring, then the node can be promoted to a higher level ring within the same band. In the event no higher Chordal ring exists within the same band, then there are two possibilities. The first is that the node's score meets or exceeds the $minThresh_i^{band}$ of the next higher sector band. In this case the node is simply promoted to the higher band. The second possibility arises in the event that the node's score exceeds the current band but still does not meet the $minThresh_i^{band}$ of the next band. If this is the case, then the node can either remain within its current position until its score rises (*overtime through increase in stability, or by increasing QoS of its provided services*) or a band reorganization process takes place. In the latter case, the MP under consideration forms a new Chordal ring with higher $minThresh_i^{band}$ and $maxThresh_i^{band}$ levels than the current and promotes a subset of the existing MPs into this newly formed ring.

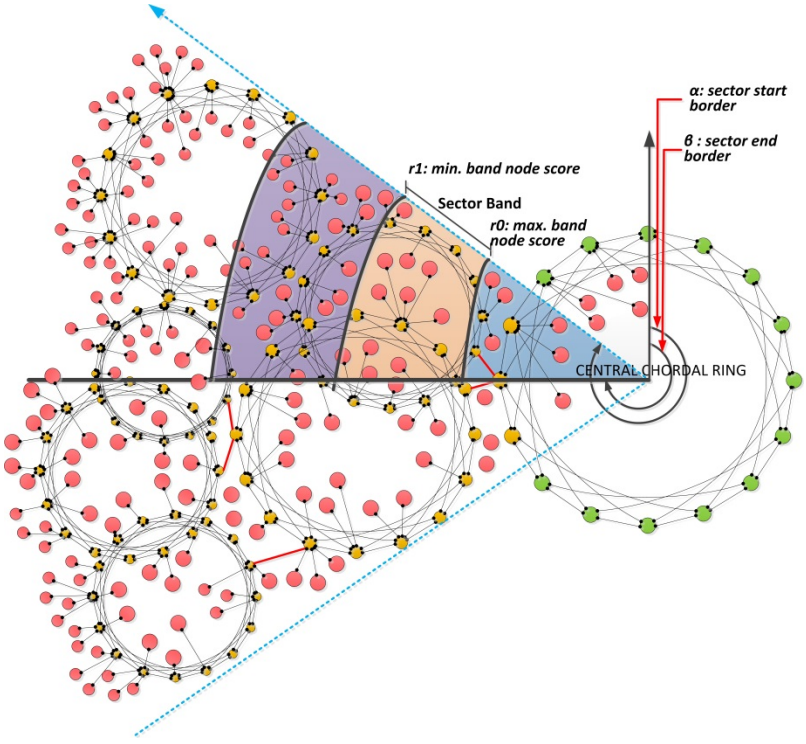


Figure 4.7 QoS-based sector and band division of the hybrid service overlay network.

Promotion in the SON continues until the central Chordal ring is reached. Nodes exceeding the central Chordal ring's thresholds begin forming a new higher level Chordal ring to become the new center of the H-SON. Figure 4.7 illustrates a partial view of the H-SON. It shows the presence of the two main types of nodes in the system; major nodes and minor nodes. Illustrated is a view of the central Chordal ring with a partial view of one of the service-type-based sectors. The figure also provides an abbreviated view of the typical sectorial bands present within an H-SON sector.

4.4.3.3. Dynamic Topology Reconfiguration

As discussed earlier, each Chordal node is capable of supporting links to a number of child nodes (both *leafs* and *LM* nodes). The number of sustainable child links of a major node are dependent on internal and external factors. Internal factors includes the amount of available resources such as processing power and memory. External factors reflect a controlling variable set by the management layer. The variable is set such that a balance is maintained between the number of structured and structured nodes. As such, the variable is chosen with the number of nodes currently present in the network taken into consideration, and to control the network recovery delay after a node failure, reduce the number of service query messages, and to reduce SSON composition delay.

Consequently, a *Major* MP receiving a join request when it has reached its maximum number of permissible child links can take two paths. The first is by rejecting the join request and forwarding it to a neighboring Chord node for possible acceptance. The second is for the receiving major node to reorganize its minor nodes. This process is triggered by the LM node contacting its minor nodes which have not become part of a lower Chordal ring. In other words, only leaf nodes and no *Major Link* (LM) nodes are contacted in the process. The child leaf node with the highest score is determined from the group. That node is sent a *FormNewPartition* call message informing it to start forming a new lower Chordal ring. This node becomes an LM node linking the parent node with the new lower level Chordal ring. The new LM node becomes the only child node from amongst the affected group while others become neighbors within the same Chordal ring. Figure 4.8 visually illustrates the formation of lower H-SON partitions as new MPs join the overlay.

A *JoinNewPartition* call message is sent to all other minor nodes informing them of

joining the new Chordal ring. The message includes the identity of nodes that must be contacted, i.e. the new LM node. The next step involves sending a join acceptance message to the MP node that initialized join request. This process of regrouping single *leaf* nodes is continued by the LM node until all of its children have become LM nodes themselves of lower Chordal rings.

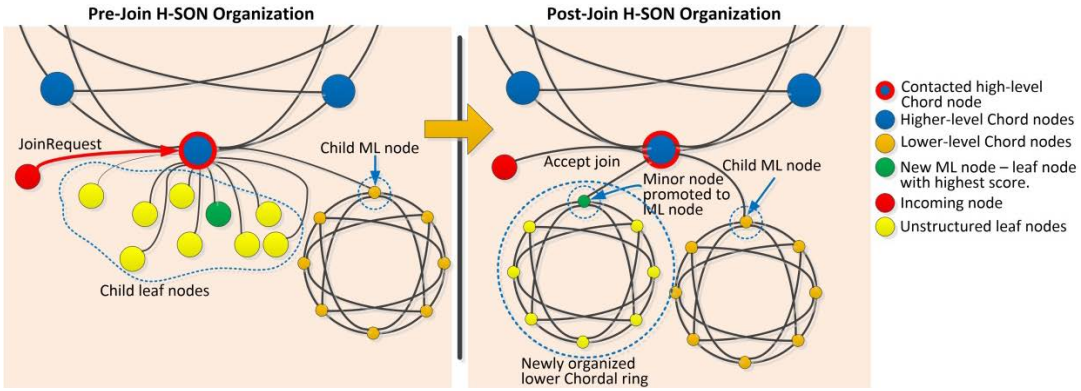


Figure 4.8 Re-organizing Minor Nodes into sub-Chord topology to accommodate further node joins in the hybrid service overlay network.

4.4.4. Handling Node Failures

The H-SON was designed specifically to minimize the effects of node failures on the overall integrity of the overlay topology. It was also designed to minimize the number of messages exchanged to reconnect the overlay once links have broken and to reduce the number of messages needed to stabilize and heal the network. Therefore handling node failures depends on whether the failure is of a minor or a major node.

Since each minor node is only linked to its parent *major* node, failure of *minor* nodes does not negatively affect the overlay structure. The affected *major* node simply removes any references in its routing table to the failed node and can accept additional join requests from arriving MPs. Failure of *major* nodes is categorized into failures of *LM* nodes and *NM* nodes. Failure of an *LM* disconnects its personal Chord structure from the parent major node and breaks the uniform shape of the Chordal ring. As a result, reconnecting the network has two phases. If the child node is a *major link node*, then failure of the parent node means separation of the lower level Chord structure and all sub-structures attached to it from the

higher Chord structure that contained the parent node. In the first phase, the Chord structure detects failure of one of its nodes. All affected Chord nodes reconfigure their links to restructure the Chord network and form all links necessary to stabilize the network. In the second phase, from the level of the highest detached Chord structure, one node must be chosen as the new *Major Link node*. This node will connect with the same *major node* the old *link node* was connected to.

The process of choosing a new *link node* is done by a search process performed to find the node with the highest score. During the lifetime of a Chord network, nodes periodically exchange stabilization messages to update successor, predecessor, and finger table links. We utilize these messages and expand them to include updates for node scores. Each node part of the optimal Chord structure maintains a table that includes node scores for its immediate successor, predecessor, and two finger links. Therefore, to discover the node with the highest score it would take at most *log n* messages. This process starts at any random node in the reconnected Chord structure. The node evaluates, from the four nodes it is connected to as well as itself, which of these nodes has the highest score. A message is sent to the node's successor containing the identity of the chosen node and identities of rejected ones. The contacted node then performs the same process and passes the message to its immediate successor. This process will continue until all nodes within the Chord structure or one of the nodes connected to them is informed. The node with the highest score is then informed of being chosen as the new *major link node* of the structure. The chosen node sends a join request message to the previous *link node*'s parent to reconnect the detached Chord structure and its sub-structures.

4.4.5. Sectorial Divisions of H-SON and the Circular MP Space

In an earlier work presented in [32], search for required MPs to form a service composition was performed based on three assumptions. First, the location of the MC requesting a media stream is known. Second, the location of the MS; the source of the media stream, was also known. Third, the search for MPs was performed in the direction emanating from the MS towards the MC only. Using the above, any node receiving a composition query forwards the query to local nodes only if it falls within an α degree

search scope. The angle α represents the maximum search scope for possible component services. This value depends on the locations of the MC and the MS and the likelihood of successfully finding valid MPs within α. This approach in general, avoids sending queries to regions of the network where answers are not likely to be found.

However, in wireless mobile networks, the dynamic movements of nodes at the network layer may result in an unequal distribution of MPs at the overlay level. We may end up with high density regions where large numbers of diverse MPs exist while other regions may lack any MPs or have them sparsely located. In the latter case, it is highly likely that α must be continuously increased until all necessary component services are found. Consequently, the delay experienced and the number of probed nodes increases with each iteration of α expansion.

In our work, we overcome the limitations of the previous solution through two methods. First is by providing a hybrid service overlay network topology structure, Section 4.4. Second is by dynamic node promotion and demotion according to node scores in (4.2). These two methods result in what can be described as a circular MP space that is further re-organized based on the types of services provided. As a result, nodes arriving in the network must join the H-SON by determining the most-fit overlay sector and band to which they belong. The successful completion of this process is based on two requirements:

- The center point of the H-SON is a known region that can be located by all nodes joining the network. It is a hypothetical virtual midpoint of the overlay. The initial contact with the H-SON management layer informs the joining MP about the H-SON's organization. This includes the overlay sectors' boundaries and the position of the overlay's center point.
- Angles α and β representing the start and end borders, respectively, of each service-type sector are also well known by all nodes joining the network. α is measured from the absolute 0^o angle pointing to a relative direction accepted by all nodes in the MP space. This information is acquired from the overlay management layer during the join process. As the number of nodes in the overlay and the variation in node densities of the respective service sectors increase or decrease, the management layer adjusts sectors' borders and informs all affected MPs.

The location of the H-SON center point and the angular borders of each service sector are controlled by the management layer. These variables depend on the total number of MPs present within the network as well as the total number and variation in services provided by these MPs. Using the above two assumptions, any search for a specific type of service to forum a composition can be done by searching within specified regions in the MP space. A simple example may involve a composition plan that requires a video encoding conversion service at some point along the path. The service required may possibly decode an MP4 video input and produce an AVI encoded video stream. The MP space sector that provides this '*type*' of service is to be located. By 'type' we assume the general operation of the service. In the above case, type refers to "encoding conversion" in its most general case. The details of the required service are dealt with in the subsequent steps of the search process. Possible angular borders for the above example may be starting at $\alpha=47°$ away from the absolute 0º angle of the MP space, with a total angular range of $\beta=35°$. This region represents the total H-SON space within which media coding conversion services may exist regardless of the QoS required to meet the MC's composition request conditions. Additionally, the specified region is not specific to the exact inputs and outputs of the required service, nor is it specific to nodes with stability levels acceptable for the composition. Stability in our case refers to the length of time a node has been present in the network and thus the length of time this node is likely to remain present in the future.

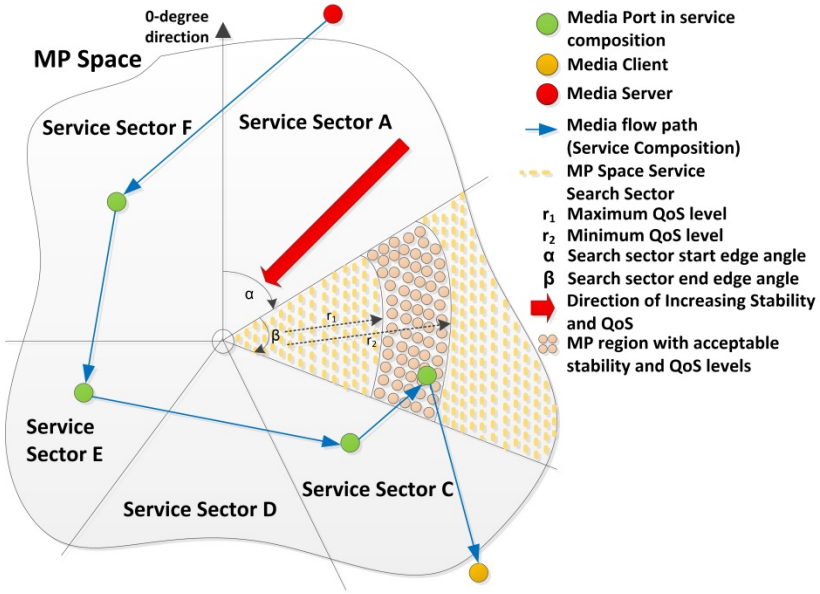


Figure 4.9 Potential MP search areas in circle-like sectors.

Consequently, to further reduce the virtual search area for potential component services we have indicated that the dynamic promotion and demotion of overlay nodes in the H-SON results in movement of nodes with higher scores towards the center of the network, while those with degrading scores are pushed towards the outer edges of the overlay. Therefore, nodes with higher levels of QoS, stability, provide popular services, etc… are generally located closer to the center of the MP space. We can utilize this overlay organization to search for nodes that meet the MC's service quality. This is done by finding a QoS upper and lower thresholds which, when accumulated for each component service in the composition would maintain an acceptable level of QoS for the MC. In Figure 4.9, r_2 represents the maximum distance away from the network's center where a node having the minimum acceptable level of stability and QoS can be chosen to perform the component service required for the composition. Conversely, r_1 represents the distance away from the center of the network where nodes with the maximum level of stability and QoS exist. Nodes located in this overlay region meet the minimum QoS requirements of the MC's service composition. Incorporating nodes in the region with a radius less than r_1 is possible, however, it would mean a lower level of resource utilization for chosen nodes. From the above, we can conclude that the overlay's searchable geographical area for a possible MP in a composition is given by the following equation:

$$SearchArea = \frac{\pi\beta}{360}(r_2^2 - r_1^2) \tag{4.4}$$

where β represents the component service's sector angle, r_2 represents the area's border with the lower acceptable level of QoS, and r_1 represents the area's border with the highest expected QoS level. Note that the same process is usually applied when an arriving MP is attempting to join the H-SON. Any node that receives a query can then be added to a composition if the following conditions from (4.5) can be met:

$$\left(\alpha \leq \theta \leq (\alpha + \beta)\right) \; and \; \left(D_{Low} \leq D_{MP} \leq D_{High}\right) \tag{4.5}$$

where Θ represents the MP's angle relative to the absolute 0º angle known by all nodes and D_{Low} represents the distance of any point on the arch r_1 distance away from the center of the MP space given by the following formula:

$$D_{Low} = \sqrt{(x_l - x_c)^2 + (y_l - y_c)^2} \tag{4.6}$$

where x_l and y_l are the distances on the x- and y-axis of any point on the lower arch, and where x_c and y_c are the (0,0) point representing the MP space's center. The same can be applied to D_{High} by replacing x_l and y_l with x_h and y_h respectively. Figure 4.10 presents a visualization of the bounded searchable MP sector band of a H-SON during a service query.

Using the above approach, we can rapidly decrease the search time for component services to form an SSON composition and to reduce the number of queried MPs in the process. Through our H-SON, the service sector angle of search is reduced to an area guaranteed to contain the type of service required for a given composition. In addition, the use of node scores for node positioning in the overlay, relative distances of the required MPs can be utilized to further decrease the number of potential nodes for use in the composition.

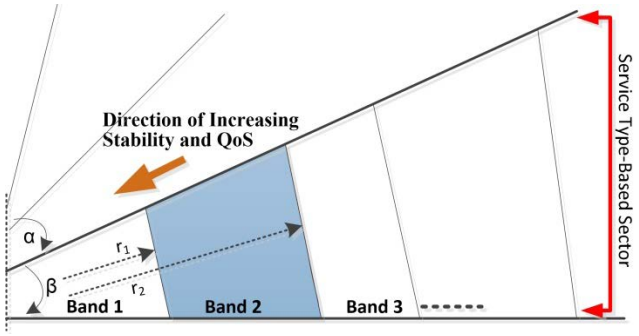


Figure 4.10 An illustration of the MP sector band to be searched bounded by (α_i,β_i,r_1,r_2).

4.4.6. H-SON Algorithms

In this section we present the algorithms used to form our H-SON and maintain its integrity in dynamic mobile environments. We also demonstrate how the H-SON can be efficiently used for MP service discovery during SSON compositions.

4.4.6.1. Joining the Central Chordal Ring

Joining the central Chordal ring begins with an arriving MP (initiator) sending a *JoinRequest* message with a *TTL* timer. A successful join process requires an acceptance reply before the timer expires. The request is sent to the nearest central Chord member the MP can contact. The expiration of the timer requires searching for a replacement Chord member to contact.

If none are found, a second attempt to the same MP is made. The receiver of the join request performs an evaluation of the current system central Chord threshold that determines whether maximum capacity in terms of number of members has been reached. If the threshold has been reached, the join process expands by creating lower levels of connections discussed in the next section. However, if the number of nodes is below the threshold, then an appropriate join sector; $(\alpha_{sender}, \beta_{sender})$, is evaluated. The join request is forwarded to an appropriate MP and an acceptance is sent back to the incoming node with information detailing its new join region. Predecessor and Successor links are reconfigured, and reconfiguration of the Chordal finger links is performed over the Chord ring once the node joins. Figure 4.11 provides a detailed algorithm of the central Chord join process, while Figure 4.12 provides a simplified flow chart for the algorithm.

The presented algorithm utilizes a TTL variable to monitor the expiration of join requests. To take into account possible network delays, the join process is preceded by a Ping-Pong exchange of messages between the joining MP and the contacted Major node. During this exchange, the joining MP evaluates the experienced network delay. The TTL value evaluated used in all subsequent messages would now take network delay into consideration.

Figure 4.11 Algorithm 1, joining central Chord in H-SON.

Algorithm 1: Joining Central Chord in H−SON

```
  Node N is a MP joining the network.  It must find its appropriate attachment location in the overlay.
  Status := {INITIATOR, WAITING, DONE, IDLE}
  Sinit := {INITIATOR}                                          %starting state of the initiator MP.

−−−−−−−
INITIATOR
  Spontaneously
  {       Send JoinRequest(SDesc, SQoS, TTL) to nearest MP in H−SON        %send join request.
          become WAITING}                                                  %wait for response.
          If (timer (TTL) expired) {          %timer has expired, begin searching for another MP to contact.
                  Search for contactable MPs.
                  send JoinRequest(SDesc, SQoS, TTL) to nearest MP in H−SON
          }become WAITING
          Receive (joinResponse(response));                                %received join response.
          If(response=accept) {
              predecessor= joinResponse.getSrc();                          %assign responding nod as pred.
              successor= predecessor.getSuccessor();                       %assign the successor node.
              finger = ((initiator.id()+c) mod totalNodes);                %assign all finger links.
              become IDLE; } }                                %join has completed, enter the idle state.
WAITING                                                %waiting state, wait for incoming join response.
  if T>TTL {                                                             %Has the timer expired yet?
          resend JoinRequest;                                  %timer has expired, resend the join request.
  }
−−−−−−−
RECEIVER                                                %Starting state of the joinRequest receiver.
  Receive (JoinRequest(SDesc, SQoS, TTL))                        %Join request received from MP.
```

[CONTINUED ON P.80]

[CONTINUED FROM P.79]

```
If (totalNodes < C_Thresh) {          %maximum number of supported nodes in central Chord has NOT been reached?
      Use S_Desc to evaluate (α_sender, β_sender)                    %find the MP' s potential H-SON sector.
      if ((α_receiver ≈ α_sender) && (β_sender ≈ β_receiver)) {      %Receiver and sender fall within same sector?
          %Service type of sender falls within same sector as the receiver.
          send (joinAccept, predecessor, successor);                 %send a join accept as part of central Chord.
          totalNodes++;                                              %increment total nodes in central Chord.
          successor = JoinRequest.getSrc();                          %set sender as my successor, fix fingers.
          finger = (receiver.id()+c mod totalNodes);
          become CONNECTING;   % forming Chordal links    }
      Else {                                                         %arriving node should not be in my sector.
          %Retrieve next neighbor closest to service sector.
          nextHop = one of {successor, predecessor, fingerNode1, fingerNode2};
          forward (JoinRequest(S_Desc, S_QoS, TTL), nextHop);        % forward join request to appropriate region.
          become IDLE;   }
  }
  else if (totalNodes ≥ C_Thresh) {                                  %max. number of supported children reached.
        begin forming next level (Algorithm 2).                      %Begin forming a lower level Chord ring.
        if (second level formation successful){                      %Was able to form a new lower Chord ring?
            send (joinAccept, Leaf);                                 %send an acceptance as a leaf node to the replier.
        }
        else{                                                        %if the second level Chord formation was not successful.
            nextHop = one of (successor, predecessor, fingerNode1, fingerNode2);   %find my next hop MP
            forward (JoinRequest(S_Desc, S_QoS, TTL), nextHop);      %forward the join request to next hop.
            become IDLE;                                             %sleep.
}
```

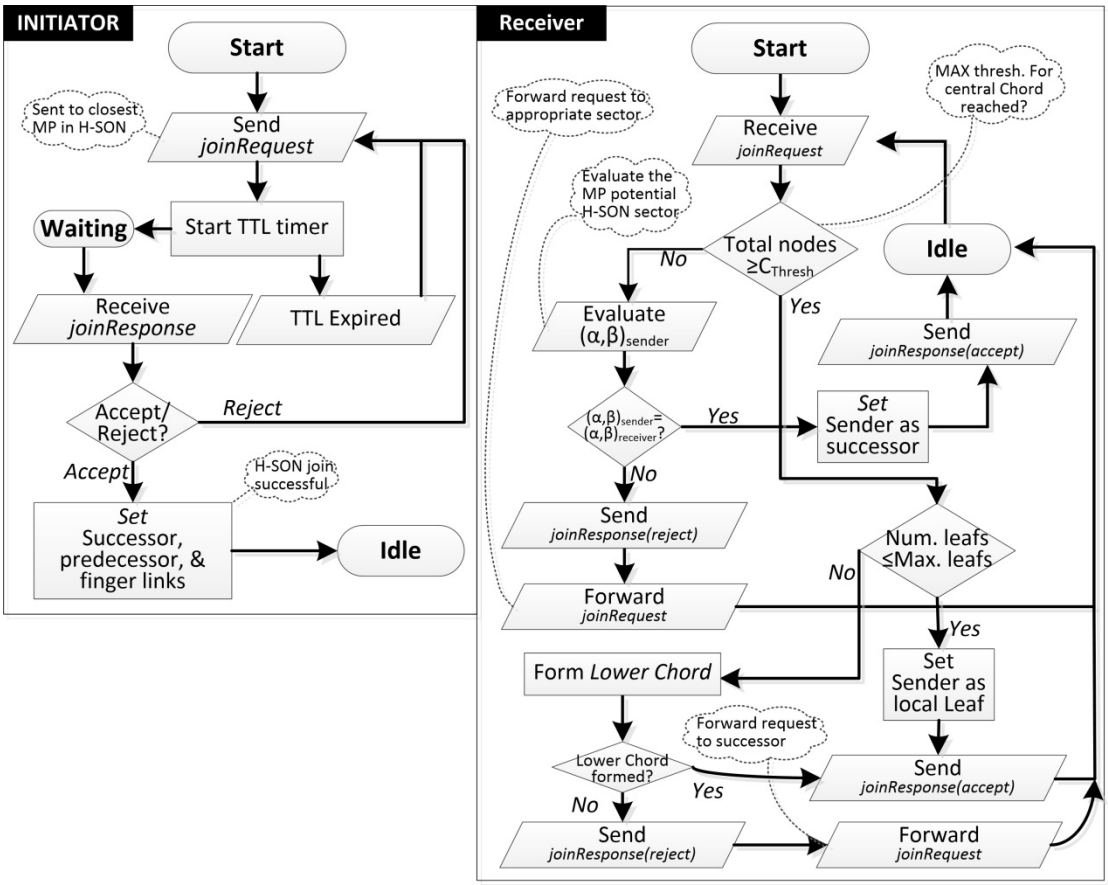


Figure 4.12 Flow chart for a simplified overview of Algorithm 1, joining central Chord in H-SON.

4.4.6.2. Joining Lower Levels in H-SON

Once the C_{Thresh} of the central Chordal ring has been reached, incoming MPs join various hierarchically lower Chordal regions in the H-SON. Joins can occur either as leafs to major nodes, or as neighbors to major nodes. The placement of incoming MPs depends on their nodes' scores (4.2). As such, much of the second algorithm is shared with the previous one. However, in addition to the sector search performed previously, the receiving MP now performs an evaluation of the sector band to which the incoming MP's join request should be forwarded. Evaluation is performed based on the MP's quality of provided services S_{QoS}^{MP} by evaluating all QoS properties of these services: $QoS_{property[i]}$. Thus the minimum and maximum distances away from the H-SON center; *(r1, r2)*, where the MP join request must be forwarded to is found.

However once the join request has reached its potential destination, the receiving MP may not be able to accommodate the incoming MP due to lack of resources or because it has reached its maximum permitted number of child nodes (this is the case when the incoming MP must join as leaf, i.e. not a major node). In such event, the child node reconfiguration process discussed in Section 4.4.3.3 is performed by forming a new partition. This process is detailed in Algorithm 1. The process by which messages such as join requests are forwarded in H-SON are also detailed in Algorithm 2 of Figure 4.13. A flow chart of Algorithm 2 is presented in Figure 4.14.

Figure 4.13 Algorithm 2, joining second-level Chord in H-SON.

Algorithm 2: Joining Second Level in H−SON

```
Node N is a MP joining the network. The maximum threshold permitted for central Chord has been exceeded.
MP must find its appropriate attachment location in the overlay.
Status := {INITIATOR, WAITING, DONE, IDLE}
Sinit := {INITIATOR}                                   %starting state of the initiator MP.

−−−−−−−
INITIATOR
  Spontaneously
  {     Send JoinRequest(SDesc, SQoS, TTL) to nearest MP in H−SON      %send join request.
        become WAITING}                                                %wait for response.
         If (timer (TTL) expired) {     %timer has expired, begin searching for another MP to contact.
            Search for contactable MPs.
            send JoinRequest(SDesc, SQoS, TTL) to nearest MP in H−SON
         }become WAITING
        Receive (joinResponse(response));                              %received join response.
        If(response=accept) {                                          %if the join request has been accepted.
```

[CONTINUED ON P.82]

[CONTINUED FROM P.81]

```
            predecessor= joinResponse.getSrc();                          %assign responding nod as pred.
            successor= predecessor.getSuccessor();      %my predecessor's successor is now my successor.
            finger = ((initiator.id()+c) mod totalNodes);                %fix my finger links.
            become IDLE; }
}
WAITING                                                  %waiting state, wait for incoming join response.
  if T>TTL {                                                          %Has the timer expired yet?
        resend JoinRequest;                                %timer has expired, resend the join request.
  }

-------
RECEIVER
  Receive (JoinRequest(S_Desc, S_QoS, TTL))      %receive join request and determine if central thresh. is reached.
  Assert (totalNodes ≥ C_Thresh);
  Use S_Desc to evaluate (α_sender, β_sender);
  if ((α_receiver ≤ α_sender) && (β_sender ≤ β_receiver)) {                %MP's service falls within my sector?
      if (receiver.getType() == leaf) {                %receiver is a leaf node.  Must forward to parent MP.
          --TTL;                                                    % decrement the message's TTL.
          send forwardedMessage(JoinRequest(S_Desc, S_QoS, TTL), parentNode);      %forward message to parent.
          become IDLE; }
      else if (receiver.getType() == major) {                %receiver is a major node, begin evaluations.
          %Evaluate MP QoS and determine sector band.
          for (i=0 → i=S_QoS.getNumProperties()) {                        %loop through all QoS properties.
            S_QoS^MP ← QoS_property[i]                                    %Get total QoS of MP.
          }
          get (r1, r2) from S_QoS^MP;                    %Get potential sector band borders from QoS values.
                                                                  % Check if receiver is within sector band.
          if ((r1^receiver ≤ r1^sender) && (r2^receiver ≥ r2^receiver)) {          % receiver has higher score.
            %Check my level of resources and number of attached children
            if (resources_receiver^available ≥ resources_sender^required) {      %Available resources exceed those required.
                if (children.getTotal() < children_Thresh) %Total num. of children connected doesn't exceed thresh.
                      send (joinAccept, status=child, parentNode);          %accept join as a child node.
                else if (children.getTotal() ≥ children_Thresh) {     %If I have reached my max num. of child MPs.
                  if (∃ child.getType( ) = leaf && child has highest resources) {
                      %If I have leaf children and at least one of them has respectable amounts of resources.
                      FORM_NEW_PARTITION;   (Algorithm 3)          %Begin forming sub-Chord partitions.
                      send (joinAccept, status=child, parentNode);          %Accept child as a leaf node now.
                  }
            }
            else if (resources_receiver^available < resources_sender^required) {     %not enough resources available for incoming MP.
                forward(S_Desc, S_QoS, TTL);                          % (Algorithm 4) Forward the message.
          }
          else if ((r1^receiver < r1^sender) && (r2^receiver ≤ r2^receiver)) {     % receiver has higher BAND than arriving MP.
            forward_LowerBand (S_Desc, S_QoS, TTL);              %(Algorithm 4) Forward message to lower band.
          }
          else {                                                          %receiver has lower band.
            forward_HigherBand(S_Desc, S_QoS, TTL);                %forward to higher band (Algorithm 4).
          }
      }
  }
  else                                                %arriving node belongs to a different sector than mine.
      forward [Dest(α_sender, β_sender), S_Desc, S_QoS, TTL];                  %forward to appropriate sector.
```

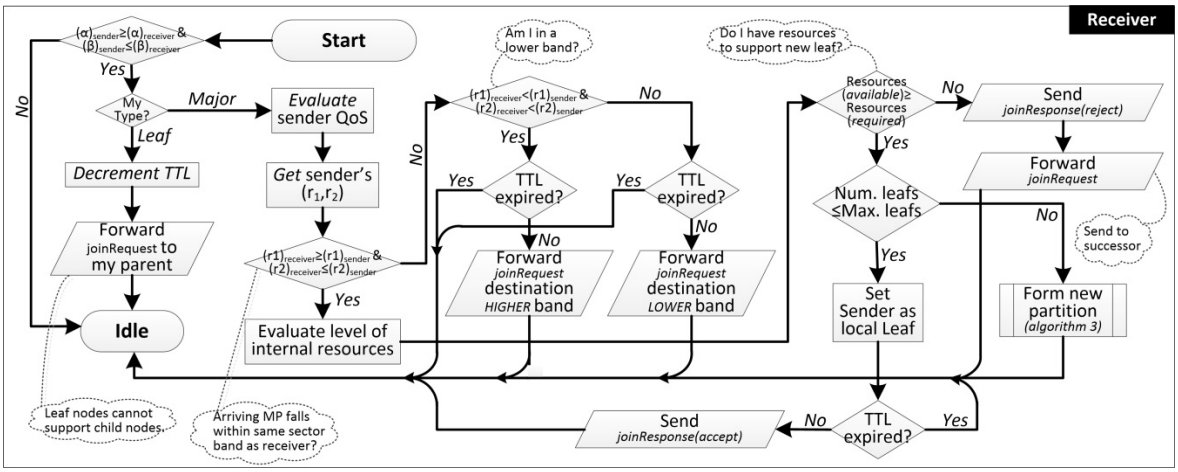


Figure 4.14 Flow chart summarizing Algorithm 2, joining second-level Chord in H-SON. Only the receiver's operations are illustrated.

4.4.6.3. Reorganizing Child Nodes and Forming Partitions

When a new node is to be added as a leaf connection to a major node that has reached its maximum threshold of permitted child connections, then a reconfiguration process is required. This process was discussed earlier in Section 4.4.3.3. The details associated with leaf nodes' reconfiguration is further detailed in Algorithm 3 in Figure 4.15. In Figure 4.16 algorithm 3 is visualized through a flow chart. Figure 4.16 A presents the steps needed to choose a new link MP, while Figure 4.16 B presents the steps performed by the new LM node to be connected to the lower Chord node.

Figure 4.15 Algorithm 3, re-organizing H-SON Minor Nodes into lower Chord topologies.

Algorithm 3: Forming Partitions

Node N is a MP joining the network. The maximum threshold permitted for children of receiving node has been reached. Receiver evaluates whether to reconfigure child nodes or forward request to neighboring node.
Status := {RECEIVER, WAITING, DONE, IDLE}

```
-------
RECEIVER
  Receive (JoinRequest(SDesc, SQoS, TTL));   %receive a join request that cannot be handled without lower Chord.
  do formLowerChord;                          %begin formation of lower Chord ring.
  do evaluateResources;                       %Evaluate available resources after lower Chord has been formed.
  if (accept) {                               %resources now available can handle incoming MP join.
      send (joinAccept, status=child, parentNode);   %accept join of MP as a child leaf node.
  } else if (reject) {                        %If join is rejected then forward message to neighbor MP.
      forward (joinRequest(SDesc, SQoS, TTL)) to neighbor;   %Algorithm 4.
  }

-------
```

[CONTINUED ON P.84]

[CONTINUED FROM P.83]

```
procedure formLowerChord( ) {                          %Procedure to begin forming a lower Chord ring.
  for (i=0 → i=this.getTotalChildren()) {              %Retrieve the total number of child nodes attached to me.
      if (this.getChild[i].getType ≠ LM) {             %the child node must not be a link node.
          queue child[i];                              %All child leaf nodes are queued to begin forming a Chord ring.
  }}
  get child[j] from queue with highest score.          %retrieve the child with the highest score.
  send FormNewPartition message to child[j];           %tell the child MP to form a lower Chordal ring.
  for (i=0 → i=this.getTotalChildren()−1){             %inform the remaining child nodes to form Chordal
      send JoinNewPartition(child[j]);                 %ring with highest−scored−node as their LM.
  }

−−−−−−−
procedure evaluateResources ( ) {                      %Procedure to evaluate available internal resources at a MP.
  if (resources_receiver^available ≥ resources_sender^required) {   %Available resources exceed those needed?
      return accept;                                   %indicate acceptable resource levels.
  } else
      return reject;                                   %indicate unacceptable resource levels.
}

−−−−−−−
IDLE
  receive (FormNewPartition, numArrivals) {          %receive message to form new partition with number of
                                                     incoming Chordal ring neighbors;
      become ACTIVE;
      do formingPartition();                         %Call the formingPartition procedure.
  } receive (JoinNewPartition (child[j])) {          %receive message to join new partition from a child node.
      become ACTIVE;
      do joiningPartition(child[j]);    %child node begins process of joining the new partition as a Chord member.
}

−−−−−−−
procedure formingPartition( ) {                        %Begin forming a lower Chord partition.
  %maintain current connection to parent node.
  this.setType() = LM;                                 %switch node type from minor to major link.
  wait for incoming connections;                       %child nodes will arrive with a request to join partition.
  receive joinNewPartitionCall (OldParent, TTL);  %receive calls from old sibling nodes to join new Chordal ring.
  if (T<TTL) {                                         %Only accept joins if the time window has not closed yet.
      if (this is first join) {                        %First child node to join the partition.
          send joinNewPartitionReply (Accept);
          this.setSuccessor(sender);
          this.setPredecessor(sender);                 %succ and pred becomes the new arriving node
      } else {
          sender.setSuccessor (this.getSuccessor);     %my old successor becomes senders new successor.
          this.setSuccessor (sender);                  %my new successor is the sender.
          sender.setPredecessor(this);                 %I will become the predecessor of the arriving node.
          send fixFingers();                           %begin fixing the finger links.
}}

−−−−−−−
procedure joiningPartition (chosenLM) {                %attempt to join new partition.
  send joinNewPartitionCall (this.getParent(), TTL) to chosenLM;   %contact chosen LM node.
  become WAITING;
  receive joinNewPartitionResponse (Accept) → form connections.    %begin forming connections if accepted.
}
```

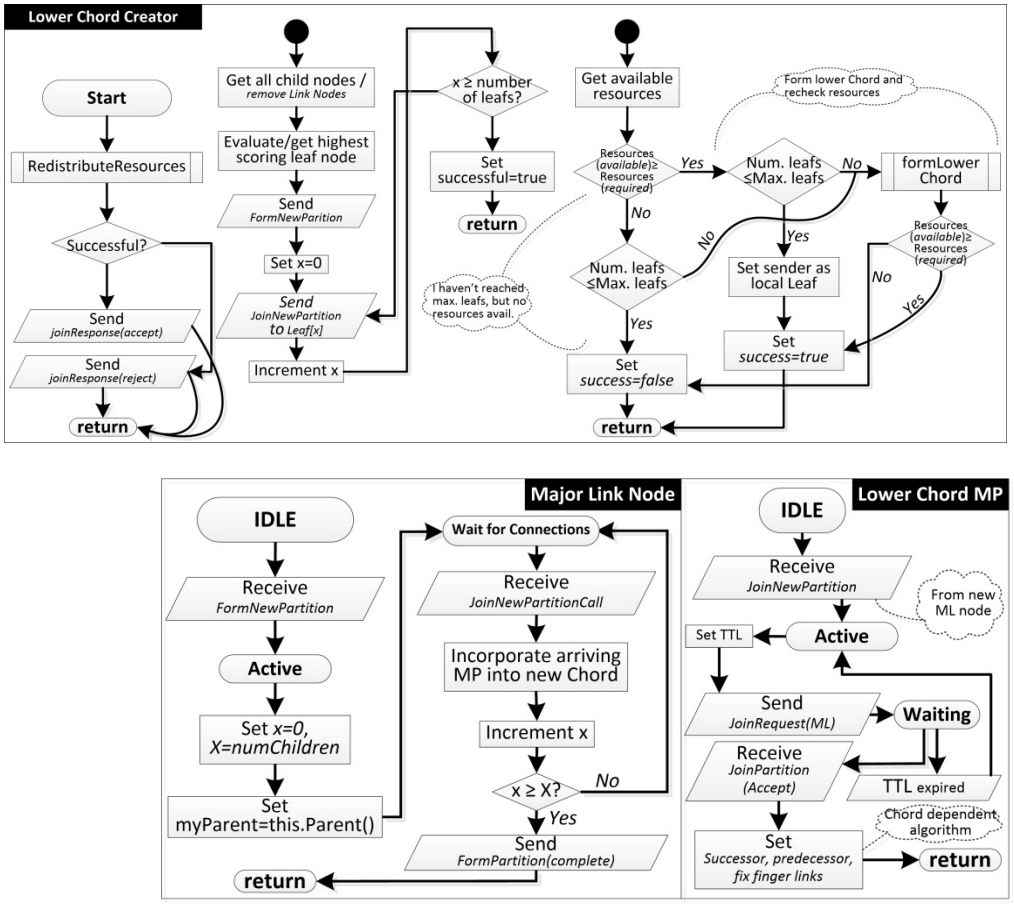


Figure 4.16 Flow chart for Algorithm 3, re-organizing H-SON Minor Nodes into lower Chord topologies. A) Steps performed by the creating MP of the lower Chord. B) Steps performed by the new LM node and the to-be connected lower Chord nodes.

4.4.6.4. Forwarding Messages in H-SON

Message forwarding is based on three conditions that must be fulfilled for each delivered message to determine the direction a message is forwarded to:

- Evaluating the service sector at which the target MP is located.
- Determining the sector band within which the target MP is located. This is determined based on the target node's score value.
- The node type of the message receiver. In the event a leaf node receives the message, it is only forwarded to its parent node. However, major nodes forward messages in four possible directions. The first two are the predecessor or successor

nodes. The third is to one of its child Major Link nodes if any exist. Forwarding to these nodes represents the sender's intention to move the request towards lower Chordal rings. The fourth direction is to a child leaf node. No forwarding occurs once a leaf node is reached.

Figure 4.17 illustrates algorithm 4 detailing the message forwarding process.

Figure 4.17 Algorithm 4, message forwarding in H-SON.

Algorithm 4: Forwarding Messages in H−SON

Message X has arrived at MP N. Decision has been made to forward the message and must determine destination of forwarding.

−−−−−−−

```
RECEIVER
  receive message(X);                                          %message to be forwarded received.
  evaluate (α destination, β destination);                     %Evaluate sector angle of destination.
  if (within same sector){                                     %Am I in the same sector?
      evaluate (r1^destination, r2^destination);               %Evaluate the band area.
      if (within same band) {                                  %Am I in the same band?
          If (resources_receiver^available < resources_sender^required) {   %Not enough resources available.
            Switch this.getType()                              %Determine my type to begin the forwarding process.
            Case LEAF                                          %I'm a leaf node.
              send message(X) to this.getParent();             %Leaf forwards message to its parent.
            Case LM                                            %I'm a major link node.
              Source=message(X).getSource();                   %LM nodes forward messages based on their origins.
              If(source=this.Parent()){                        %If message arrived from parent then:
                  If (this.Children() != Empty){               %I have no children.
                       If (this.Children() within same sector and band){   %send to children if they exist and they
                                                                           fall within same sector and band.
                          Broadcast to this.Children();        %broadcast message to my children.
                     } }
                  else {
                      send message(X) to this.getSuccessor();  %No children, send to successor.
              Else if(source=this.Child(i))                    %Source is one of node's children.
                  send message(X) to this.getSuccessor();      %Send message to successor.
              Else if(source=this.Successor())                 %Source is successor, send to predecessor.
                  send message(X) to this.getPredecessor();
              Else if(source=this.Predecessor())               %Source is pred. send to succ.
                  send message(X) to this.successor();
            Case MN                                            %I'm a major node.
              If (source=this.Successor())                     %Source is successor, send to pred.
                  send message(X) to this.getPredecessor();
              Else if(source=this.Predecessor())               %Source is pred. send to succ.
                  send message(X) to this.successor();
          }
          Else (resources_receiver^available ≥ resources_sender^required) {   %There are enough resources.
              Accept message(X);                               %Accept message.
          }
      }
      if (NOT within same band) {                              %Within same sector, not same band.
          Switch this.getType()
            Case LEAF
              send message(X) to this.getParent();             %Leafs forward messages to parents.
```

[CONTINUED ON P.87]

[CONTINUED FROM P.86]

```
            Case LM
              if (lower band){                          %LM send to children if destination is lower band, or
                  If (this.Children() != Empty){        %send to successors if no children.
                      If (∃ this.Children().getType() = LM){
                      Broadcast to this.Children() LM;  %only broadcast to children that are LMs.
                  } Else
                      send message(X) to this.successor();
              } Else
                  send message(X) to this.getParent();  %If destination is higher band, send to parent.
            Case MN                                     %MN nodes can send to children or succ.
              if (lower band){          %if the destination of a lower band, then broadcast to child nodes.
                  If (this.Children() != Empty){
                      If (∃ this.Children().getType() = LM){   %only broadcast to LM child nodes.
                          Broadcast to this.Children() LMs;
              } Else
                  Send message(X) to this.successor();
      } }
  } else if (!= same sector) {                          %Not destined to within the same sector.
      Switch this.getType()
      Case LEAF                                         %Leafs can send to parents.
          Send message(X) to this.getParent();
      Case LM or MN                                     %Majors send to closest node of its-
                                                        %neighbors that's closest to the destination.
          Get direct neighbor whose (α neighbor, β neighbor) is nearest to (α destination, β destination);
          Send message(X) to retrieved neighbor;
  }
```

The message forwarding algorithm of Figure 4.17 is simplified through a flow chart view as presented in Figure 4.18.

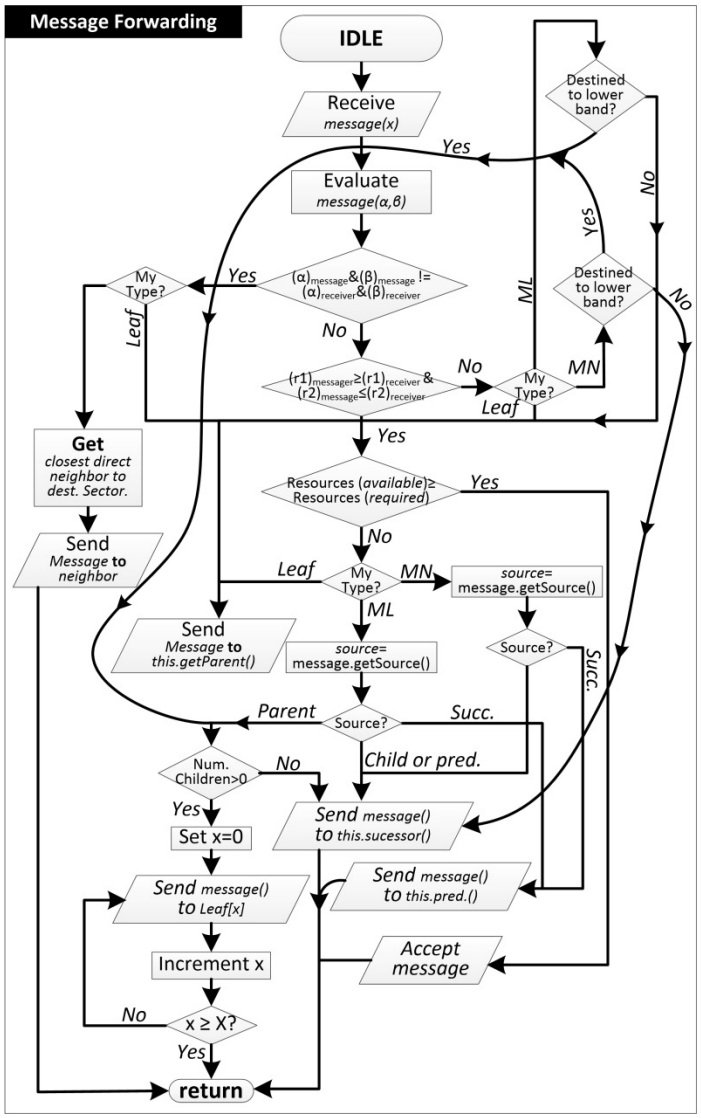


Figure 4.18 Flow chart of Algorithm 4, message forwarding in H-SON.

4.4.6.5. Promotion and Demotion in H-SON

Periodically, node scores of MPs linked in the H-SON are revaluated. This is essential in maintaining the dynamicity of our proposed overlay given its use in mobile environments. Changes in service loads, depletion of battery power, weakening of wireless signals, or other unforeseen changes in the performance of some MPs affects their abilities to join SSONs in the future and thus has direct effects on their score values. Moreover, the reverse of this is also true. Consequently, MPs whose scores decrease may need to be demoted to lower-level sector bands. On the other hand those MPs whose scores increase may be promoted to

higher bands. Promotions and demotions may occur within the same band as well according to score levels where some MPs may be promoted from a leaf to an NM status and vice versa. Promotion from an NM to an LM status occurs in the event a new MP joins the H-SON or an LM reconfigures its child leaf nodes. Both conditions were described in earlier algorithms above and will not be illustrated in Algorithm 5 in Figure 4.19, and a flow chart representation is presented in Figure 4.20.

Figure 4.19 Algorithm 5, dynamic node promotion and demotion in H-SON.

```
Algorithm 5: Promotion and Demotion in H−SON
H−SON maintains its dynamicity by periodically promoting and demoting nodes according to their current scores
−−−−−−−
IDLE
  PERIODICALLY                                                    %periodically evaluate node scores.
     set evaluationTimer = λ                                      %choose a timer period λ for evaluation.
     while (evaluationTimer NOT expired) {                        %while times has not expired yet.
        sleep for μ                                               %sleep until sleeping time is over. Sleep period set to μ seconds.
        become IDLE
     }
     become SCORE−EVALUATING;                                     %perform score evaluations of children and neighbors.
−−−−−−−
SCORE−EVALUATING                                                  %Enter the score evaluation procedure for children and neighbors.
  Get myType ← this.getType();                                    %get my type.
  if (myType == leaf) {                                           %If I'm a leaf node.
     do NOTHING;                                                  %No evaluation is required, no children or neighbors present.
     cancel evaluationTimer;}                                     %Cancel my evaluation timer.
  else if (myType == LM or NM) {                                  %If I'm major node or a link node, then..
     for (i=0 → this.getTotalChildren()) {                        %retrieve all my children to contact.
        evaluate nodeScore (Child[i], ∃S_Desc ~ this.getDesc(), ∀S_QoS)   %evaluate score based on QoS properties
                                                                  for all services falling within same sector.
        modify nodeScore according to (currentTime − Child[i].getArrivalTime());   %the longer a node stays,
                                                                  the higher its score.
     %increase score since stability has increased.
     }
     if (∃Child with nodeScore > leaf position) {                 %if the node's score exceeds its current position.
        if (Chordal Ring NOT reached C_Thresh) {                  %There is still space in Chord ring.
           send promote (successor=this.getSucc(), predecessor=this.getID())
                                                                  %promote child to my successor in ring.
           this.setSucc(Child[i]);                                %set the child as my successor.
           repeat fixFingers(); }}                                %fix all necessary finger links.
                                                                  %Evaluate neighbors for possible demotion only;
     evaluate nodeScore (successor, ∃S_Desc ~ this.getDesc(), ∀S_QoS)   %evaluate score based on QoS of services
                                                                  that fall within the same sector only.
     if (nodeScore < Ring_Thresh) {                               %node falls below ring's minimum score.
        inform ring neighbors;                                    %inform neighboring neighbors.}
                                                                  %Reach unanimous decision regarding demotion of successor;
     if (decision == demote){                                     %if the decision is to demote the node.
        send demote to this.getSuccessor();                       %demote the successor.
        this.setSucessor() =(this.getSucessor()).getSuccessor();  %set a new successor node.
        %my old successor's successor is now my successor
        repeat fixFingers();
     }else
        Do nothing.}}
```

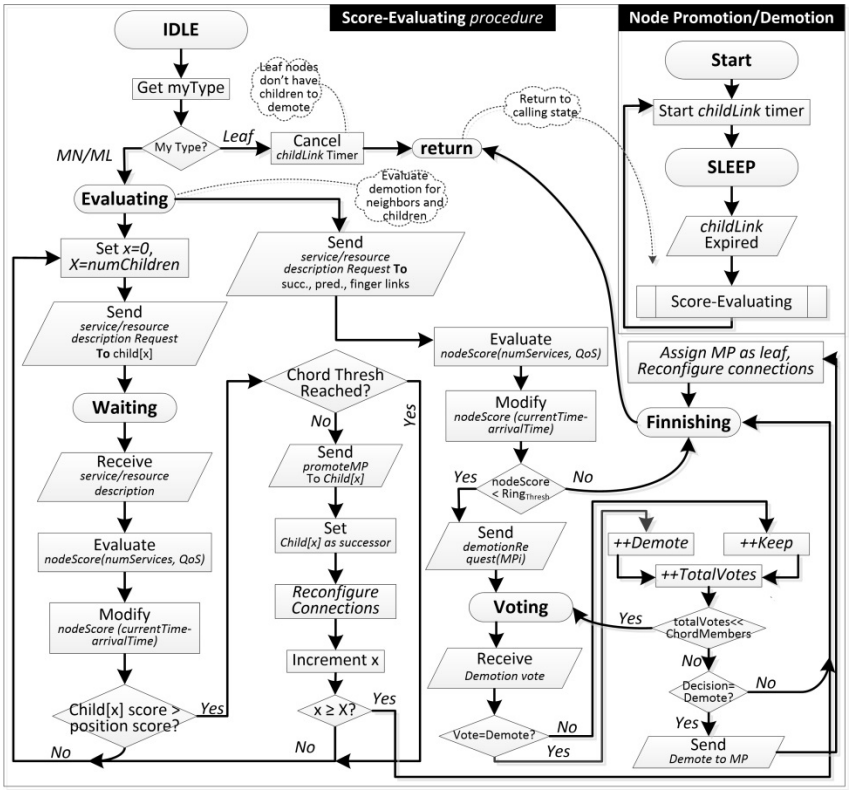


Figure 4.20 **Flow chart of Algorithm 5, dynamic node promotion and demotion in H-SON. Diagram begins at the Node promotion and demotion and score evaluation procedure.**

4.4.6.6. Leaving The H-SON

MPs that are part of the H-SON may leave without any notification such as the case when a sudden loss of connection occurs. In such event, the overlay destabilizes temporarily due to the possible breakage of one or more Chordal rings. However, all nodes (except for minor leaf nodes) should be reachable due to the overlay's multi-path structure. This error requires fast correction to stabilize the network and any established SSONs. Moreover, the possibility arises when a node informs its neighbors of its impending departure. Consequently, all affected nodes can reconfigure their connections before the MP leaves. As such, no disconnection occurs, and the network is quickly stabilized. Those two possibilities are further described in Algorithms 6 (see Figure 4.21 and Figure 4.22) and 7 (see Figure 4.23). MN and LM nodes inform their child nodes of the identity of a candidate neighboring node as their parent to avoid prolonged network disconnections.

Figure 4.21- Algorithm 6, leaving the hybrid service overlay network

Algorithm 6: Leaving H−SON

```
Node N is a MP attempting to leave the network. H−SON links must be fixed and network must be stabilized.
Status := {INITIATOR, RECEIVER, WAITING, DONE, IDLE}
S_init := {INITIATOR}                                   %starting state of the initiator MP.
S_term := {DONE}                                        %MP has left and H−SON stabilized.

−−−−−−−
INITIATOR node N
 % leaving process depends on the type of MP performing action.
 if (this.getType() == leaf) {                          %If I'm a leaf node. Must only contact my parent.
   send leaveRequest(LeaveMessage, TTL) to N.getParent();   %send a leave request to my parent MP.
   become WAITING_LEAVE(TTL);          %remain connected for a short period to maintain H−SON stability.
   become DONE;                                         %Node has left the overlay.
}
 else if (this.getType() == (MN or LM)) {               %If I am a major node; normal or link.
    SIMULTANEOUSLY                                      %simultaneously send the following messages.
    send leaveRequest(LeaveMessage, TTL, pred) to succ;     %send leave request to my successor.
    send leaveRequest(LeaveMessage, TTL, succ) to pred;     %send leave request to my predecessor.
       % begin forming connections between the pred. and succ.   %pred. and succ. Begin interlinking.
    for (i=0→number of child nodes) {
          send leaveRequest(LeaveMessage, TTL, succ, pred) to child(i);   %Inform my children of my interest
                                                                          in leaving the network.
    }
    become DONE; }                                      %Node has successfully left the network.

−−−−−−−
WAITING_LEAVE(TTL)                                      %Waiting to receive confirmation response before I leave.
 while (!TTL.expired()) {                               %while timer hasn't expired.
   receive leaveResponse(Accept);
   if (leavResponse(Accept) is received){
       become DONE;                                     %Once acceptance is received, then leave the network.
   }
 }
 %timer expired, node now leaves but attached nodes may not be informed yet.
 become DONE;                                           %Node leaves anyway, unsafe departure.

−−−−−−−
RECEIVER
                                        %Receiver's behavior depends on the sender's and its own types.
 if (this.getType() == leaf) {                          %if I am a leaf node, then my parent is about to leave.
   this.setParent() = NULL;                             % my parent node has left, nullify its link.
   replacement = leaveRequest.getRecommendedNode();     %get identity of replacement node from message.
   send joinRequest (JoinRequest(S_Desc, S_QoS, TTL)) to replacement;   %send join request to replacement MP.
   become WAITING;                                      %Algorithm 1 takes control.
 }
 else if (this.getType() == NM) {                       %the receiver is a Normal Major node.
                                                        %determine type of sender node
   if (sender.getType() == child) {                     %sender is a child of the receiver.
     if (sender.getChildType() == leaf) {               % child is a leaf, i.e. has no other links.
       send leaveResponse (Accept) to sender.getID();   % send accept response to leave request.
       This.getChildLink(i) == NULL;                    % remove link to leaving leaf node.
     } else if (sender.getChildType() == LM) {          % sender is a Major link node.
       replacement = leaveRequest.getRecommededNode();  %retrieve identity of recommended replacement.
                                                        % retrieve ID of LM node to replace node leaving.
       send leaveResponse (Accept) to sender.getID();   % send accept response to leave request.
```

[CONTINUED ON P.92]

[CONTINUED FROM P.91]

```
        This.getChildLink(i) == NULL;                          % remove link to leaving LM node.
        become WAITING;
        send joinRequest () → go to Alogrithm2                 % send join request to replacement MP.
      }
    }
    else if (sender.getType() == NM)                           % leaving MP is a neighbor.
      if (sender.getID() == successor) {                       %if the leaving MP is my successor.
        this.setSuccessor() = sender.getSuccessor();       % successor of leaving MP is now my new successor.
        this.getSuccessor().setPredecessor() = this.getID();% set myself as predecessor to the new successor.
      }
      else if (sender.getID() == predecessor) {                %if the leaving node is my predecessor.
        this.setPredecessor() == sender.getPredecessor();      %fix link to new predecessor.
        sender.getPredecessor().setSuccessor() == this.getID();%fix the new pred. succ. Link.
  }
```

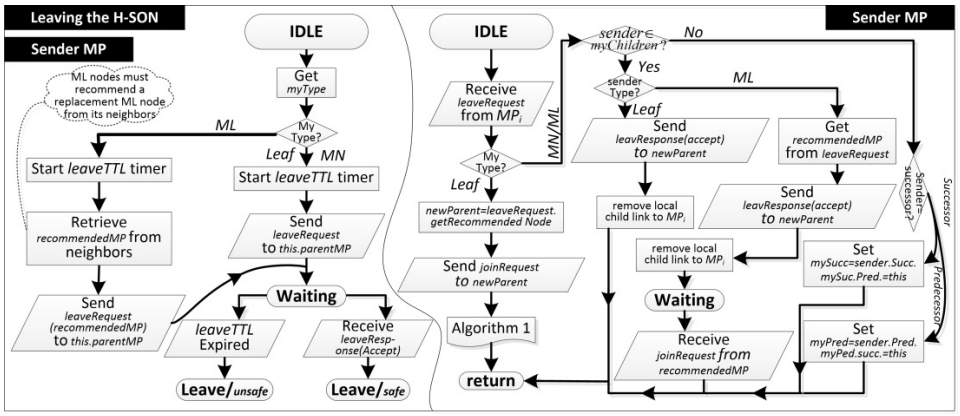


Figure 4.22- Flow chart for Algorithm 6, leaving the hybrid service overlay network.

4.4.6.7. Handling Node Failures

Figure 4.23 Algorithm 7, handling missing nodes in H-SON.

Algorithm 7: Missing Node H−SON

```
Node N is a MP that notices a connected MP in H−SON is missing
Status := {INITIATOR, RECEIVER, WAITING, DONE, IDLE}
Sinit := {INITIATOR, IDLE}                                 %starting state of the initiator MP.
Sterm := {DONE}                                            %MP has left and H−SON stabilized.

−−−−−−−
INITIATOR node N                                   %Node N periodically checks if any failures have occurred.
PERIODICALLY {
  if (this.getType() == leaf) {                            %if I am a leaf node then contact my parent.
     send LinkStabilization(stabilizationMessage, TTL) to this.getParent();   %send stabilization message
                                                                               to parent node.
     become WAITING;                                       %enter waiting state and wait for the parent reply.
  } else if (this.getType() == (MN or LM)) {               %If I am a major or link node.
       send LinkStabilization(stabilizationMessage, TTL) to this.getSuccessor(), this.getPredecessor(),
       this.getFingerLink1(), and this.getFingerLink2();   %send stabilization messages to all direct neighbors.
       for (i=0 → i=this.getTotalChildren()) {             %if I have any children, then must contact those too.
```

[CONTINUED ON P.93]

[CONTINUED FROM P.92]

```
            send LinkStabilization(stabilizationMessage, TTL) to this.getChild(i);
            become WAITING;                                          %enter the waiting state.
        }
} become ASLEEP until TTL expires or receive OK

--------
IDLE stabilization message receivers are usually in idle state.   %handling received stabilization messages.
  Receive linkStabilization (stabilizationMessage, TTL)           %stabilization message received.
    send LinkStabilization(stabilizationReply, OK) to sender;     %send stabilization response messages.
    % reschedule my stabilization message to sender.             %reschedule stabilization message to contacted MP.
    delay (stabilizationMessge, T_delay);                         %delay is specified by a predefined delay period.

--------
WAITING stabilization message sent, and waiting for reply.        %waiting for replies to my stabilization messages.
    if T>TTL                                                      %handling expiration of the message TTL timer.
      go to TTL_expired();                                        %move to TTL_expired procedure.

--------
procedure TTL_expired() {                                         %procedure for handling timer expiration.
  if (this.getType() == leaf) {                                   %if I am a leaf node.
    send JoinRequest(SDesc, SQoS, TTL) to nearest MP in H-SON     %Im now disconnected must rejoin H-SON.
    do Algorithm 2;                                               %move to algorithm 2 for joining  lower Chord rings.
  } else if (this.getType() == (MN or LM)) {                      %If I am a major or link node.
    if (dest.getType() == leaf ) {                                %one of my leafs has failed.
        remove dest.getID() from routing table.                   %remove link to the leaf node.
    } else if (dest.getType() == LM && dest.isMyChild()) {        %if failed child is an LM node.
                                                                  % destination was this node's child and is a LM node.
        send joinOffer(this.getID(), TTL) to nearest lower Chord nodes not linked to me.
                                                                  %attempting to offer disconnected lower Chord possible connection node.
    } else if (dest.getType() == this.getSucc()) {                %successor node has failed.
        send repairChord (dest.getSucc(), TTL);                   %form link with successor's successor.
        become WAITING for OK;
        receive repairChordReply(OK);                             %Chord ring links have been repaired successfully.
        this.succ.set(repairChordReply.getSender());              %form link to new successor.
        do fingerRepair(dest);                                    %go round Chord ring and fix all finger links.
    } else if (dest.getType() == this.getPred()) {                %predecessor node has failed.
        send repairChord (dest.getPred(), TTL);                   %form link with successor's successor.
        become WAITING for OK;
        receive repairChordReply(OK);                             %Chord ring links have been repaired successfully.
        this.pred.set(repairChordReply.getSender());              %form link to new predecessor.
        do fingerRepair(dest);                                    %go round Chord ring and fix all finger links.
    } else if (dest.getType() == this.getFinger1() || dest.getType() == this.getFinger2()) {
        do fingerRepair(dest);                                    %go round and fix all finger links.
    }

--------
procedure fingerRepair(failedID) {                                %procedure to repair finger links.
  repeat (repairFinger(succ, failedID));
  if (this.getSucc() [or this.getPred()] == failedID) {           %failed node is my successor.
    go to TTL_expired() do successor/predecessor repair.
  }
}
```

4.5. H-SON Algorithm Costs

The algorithms presented in Section 4.4 carry with them a complexity cost in terms of number of exchanged messages and experienced delay. Section 4.6 illustrates simulation results that evaluate the general delay experienced by nodes mobility on the general delay and rate of exchanged messages experienced by the system. As to be indicated in Section 4.6, the join delays of the H-SON are well below those of similar Chord networks. Additionally, the rate of stabilization messages exchanged (messages used to keep updated views of links) and the average rate of maintenance messages exchanged between MPs to reconnect broken links, for each node on average are also below that of Chord.

In this section we provide a detailed evaluation of the costs associated with nodes in terms of number of messages exchanged, attempting to join or leave the H-SON, in reconfiguring the H-SON topology, forwarding messages, promoting and demoting nodes, and handling failures.

4.5.1. Re-organizing H-SON Minor Nodes into Lower Chord Topology Cost

Topology reconfiguration in the H-SON occurs in the event a new MP attempts to join as a leaf node, however the contacted Major node has reached its maximum number of permissible child links. Consequently, the Major node must reconfigure its leaf nodes into a lower Chord ring and link the newly arriving MP as a leaf. Let N be the current number of child leaf node attached to Major node, MN_i. A *JoinNewPartition* message is sent to each of the N leaf nodes by MN_i. Subsequently, forming the lower Chord structure carries with it the same cost as joining an optimal Chord ring sequentially with an increasing number of nodes up to N. When a new node joins an optimal Chord ring with N nodes and K keys, then $O(K/N)$ keys must be exchanged. Consequently, the total cost of topology reconfiguration in the H-SON is given by (4.7).

$$Cost_{Reconfigure} = \left[\sum_{n=0}^{n=N} O\left(\frac{K}{n}\right)\right] + N \tag{4.7}$$

4.5.2. Join Algorithm Cost

When a new MP arrives into the network it begins by contacting the H-SON management layer with a joint request message. The delivered message includes information regarding the MP's provided service(s) and the associated quality levels. The management layer evaluates the H-SON region(s) where the MP must join virtually at the H-SON level. Every MP at the physical or network layer can be mapped to multiple virtual nodes at the H-SON layer: one for each of its provided services.

Let $MP_i(S_j)$ represent the arriving Media Port i providing j number of services such that $1 \geq j \leq J$. MP_i sends a *JoinHSON* message to the management layer and a *JoinHSONReply* message is reversely received. The MP must then send j *JoinRequest* messages to the respective H-SON regions where each of its services must join. Under the most optimistic situation, each virtual MP joins as a leaf node after receiving a *JoinAccept* message from the contacted MP. Consequently (4.8) evaluates the lowest possible join cost:

$$Join_{MIN} = 2(J + 1) \tag{4.8}$$

However, the worst case scenario arises when the MP's join requests cannot be served, and the contacted MP must restructure its child leaf nodes into a lower Chord ring. Under this situation two messages are exchanged between the arriving MP and the H-SON management layer clarifying the network sectors/bands where the each virtual image of the MP must join. Thereafter, the cost for each join becomes equivalent to a topology reconfiguration seen in (4.7) in addition to two messages exchanged between the arriving MP and the potential parent node requesting and accepting the join process. As such, the maximum cost of a join is presented as follows:

$$Join_{MAX} = J\left(\left[\sum_{n=0}^{n=N} O\left(\frac{K}{n}\right)\right] + N\right) + 2 \tag{4.9}$$

The total cost of a join then falls in the range presented in (4.10) below:

$$2(J + 1) \leq Cost_{JOIN} \leq J\left(\left[\sum_{n=0}^{n=N} O\left(\frac{K}{n}\right)\right] + N\right) + 2 \tag{4.10}$$

4.5.3. Joining Lower Chord Algorithm Cost

The cost of joining lower Chord rings consists of two messages sent to request and accept the join process in addition to the cost of exchanging keys to reconfigure necessary Chord links. Equation (4.11) evaluates the total number of messages.

$$Cost_{JOIN\_LOW} = O\left(\frac{K}{N}\right) + 2 \tag{4.11}$$

4.5.4. Message Forwarding Algorithm Cost

In this section the cost associated with message forwarding refers to the number of hops a message needs to travel to reach its destination. The minimum cost for forwarding a message is experienced when a child (Leaf) node has to send a message to its parent Major node or vice versa. The same cost is experienced when neighboring nodes in a Chord ring exchange messages. In both cases the total hop count is equal to one hop only. The maximum cost however is experienced when attempting to forward a message originating at a leaf node connected at one Chord ring that is destined to another leaf node connected to a different Chord ring in the overlay. As such, the message may need to pass through intermediate Chord rings before reaching its destination.

Let *m* represent the number of Chord rings traveled through from the source node to the destination node. Let *N* represent the number of nodes present in each Chord ring. Optimal Chord rings have only four links per node independent of *N*. Messages in each ring can be routed in $O(k)$ hops, where k is the diameter of the optimal Chord ring. For a relatively small N, $k < \log N$, while for larger values of N, $k = c \times log^2 N$. The expected range in cost for message forwarding is thus given in (4.12).

$$1 \leq Cost_{FORWARD}^{Hops} \leq m[O(k)], where \begin{cases} k < \log N, for\ small\ N \\ k = c \times log^2 N, for\ large\ N \\ and\ c\ is\ a\ variable\ that \\ increases\ with\ N \end{cases} \tag{4.12}$$

4.5.5. Promotion and Demotion of Media Ports Cost

Promoting or demoting a node begins with a query message from an MP to one of its child Leaf nodes or one of its direct neighbors in a Chord ring. A reply message is sent back by the queried node. An evaluation of the current node score is performed for the queried node to determine its current QoS levels and available resources. Accordingly a decision is made to promote or demote the node. Demoting a node results in the exchange of four messages; one message to each of the demoted MP's successor and predecessor nodes respectively, and two messages exchanged between the predecessor and successor to reconnect.

In the event of a promotion, the cost incurred is equivalent to that seen when a new node attempts to join a Chord ring with *N* connected MPs and *K* keys. However, the maximum cost occurs when the promoted node is currently a Major Link (ML) node. In this situation an additional cost of reconnecting the disconnected lower Chord ring is introduced. This is equivalent to the cost of node demotion discussed above. Consequently, the range of cost incurred in terms of exchanged messages for the promotion or demotion of an MP is illustrated in (4.13).

$$5 \leq Cost_{PROM\_DEM} \leq O\left(\frac{K}{N}\right) + 5 \tag{4.13}$$

4.5.6. Leaving the H-SON Algorithm Cost

A leaf node leaving the H-SON does not exert any effect on any nearby MPs other than its direct parent Major node. Only two messages are required; a *LeaveRequest* sent to the parent node, and *LeaveResponse* reply message signaling the acceptance for the node to leave the network. A Chord ring node without any child Leaf nodes attached to it can leave with a cost of four messages at most; two messages informing its predecessor and successor of its imminent leave, and two messages by the successor and predecessor to reconnect the broken link.

However, the cost for a Chord ring node with *m* child nodes to leave is higher. Four messages are required to inform and reconnect the predecessor and successor nodes. In

addition, *m* messages are sent to inform its child nodes about its departure. In turn, each child node must join a nearby Chord ring MP as a leaf node. Each child connects through two messages; a request to join and a response of acceptance. The total cost for an MP leaving the H-SON is presented in (4.14).

$$2 \leq Cost_{Leave} \leq 3m + 4 \tag{4.14}$$

4.6. Simulation Results

In this section, we present the evaluation tests we performed regarding the proposed lower overlay network structure using the C++-based OMNET++ [111] discrete event network simulator. This simulator permits the design of simulation models utilizing a hierarchical architecture of a system module and its various sub-modules that communicate via messages, gates, and links. An open-source overlay and P2P network simulator OverSim [112] for OMNET++ was used to model Chord overlay networks and we extended the simulator to model our proposed lower-overlay hybrid network simulation for the MP space. OverSim has a graphical user interface that illustrates the overlay and underlay topology and all network packets exchanged between modules with a central module for gathering and processing statistics.

In the performed tests, nodes were placed in a 2-dimensional Euclidean space where the delay between any two nodes was assumed to be proportional to the distance between them. In addition, a lifetime based churn model of exponential distribution was used to model the arrival and departure of nodes in the network. A sample application program of simple request-response type messages utilizing a full-recursive KBR protocol was also used. KBR encapsulates messages and forwards them to the next hop according to a local routing table of the node that is closest to the node with the destination key. The message is then forwarded recursively back to the originator.

In the simulation scenario, control messages' packet sizes depended on the type of message exchanged with a base message length of 32 bytes for join request messages. Each node was permitted a maximum of two join retries with an interval of 10 seconds before a join was considered as a failure. To maintain updated KBR application test messages were

set to a size of 100 bytes. Messages were generated at a 60 seconds interval and a failure latency of 10 seconds. Chord stabilization messages were set with the following intervals: Chord finger reconfiguration with 120 seconds, predecessor update with 5 seconds, and stabilization with 20 seconds.

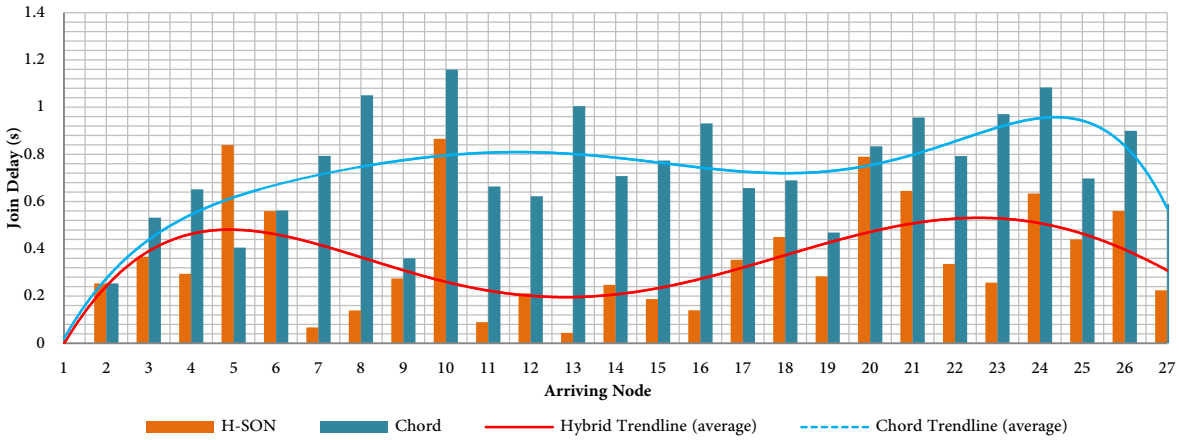


Figure 4.24 Join delay experienced between normal Chord overlay vs. proposed hybrid SON.

Implementing the proposed hybrid overlay required modification to the routing protocol used by each node from the regular chord protocols, and introducing new messages for *Major* and *Minor* node link stabilization and update, as well as link reconfiguration to form new low-level Chord structures and/or *Minor* nodes.

Given that the H-SON presents an extension of optimal Chord networks, simulation tests were performed to evaluate the benefit the presented hybrid overlay brings to SSON composition in comparison to Chord networks. Chord is characterized by its simplicity, provable correctness, and performance. Thus proving the efficiency of our overlay with respect to Chord validates its desirability over other structured and unstructured topologies. In the first scenario, two simulation tests were performed to determine the average delay experienced by each node attempting to join the network in the case of a normal Chord network compared with that of the proposed hybrid overlay. An initially empty network was used with a 10 sec. node inter-arrival rate. Results shown in Figure 4.24, illustrate a visible difference in the delay experienced by nodes in the two networks. Joining a Chord structure with N nodes and K keys requires O(K/N) keys to change hands and thus may result in some

delay before a join is completed. The same applies to the hybrid overlay, however, the existence of leaf-like unstructured extensions in the proposed overlay results in a significant decrease in the time required for a join to complete since only a *major node* is contacted to complete the join process.

Figure 4.25 presents a comparison of the mean rate of stabilization messages sent between nodes in the two networks. In Chord, with increasing numbers of nodes, the number of messages needed to maintain links to the successor, predecessor, and finger links also increases until it plateaus to approximately 30 Bytes/sec. With the same number of nodes, the same stabilization messages are exchanged in the hybrid overlay. However, the reduced number of nodes, present within each Chord structure and the distribution of the other nodes in the leaf positions indicates a decrease in the number of stabilization messages sent by each node. Even when considering the fact that our hybrid overlay introduces two stabilization messages, *Parent_Live_Stabilization* and *Leaf_Live_Stabilization* exchanged between *minor* and their controlling *major nodes* respectively, the mean number of stabilization messages sent by nodes in the overlay remains well below that of normal Chord networks.

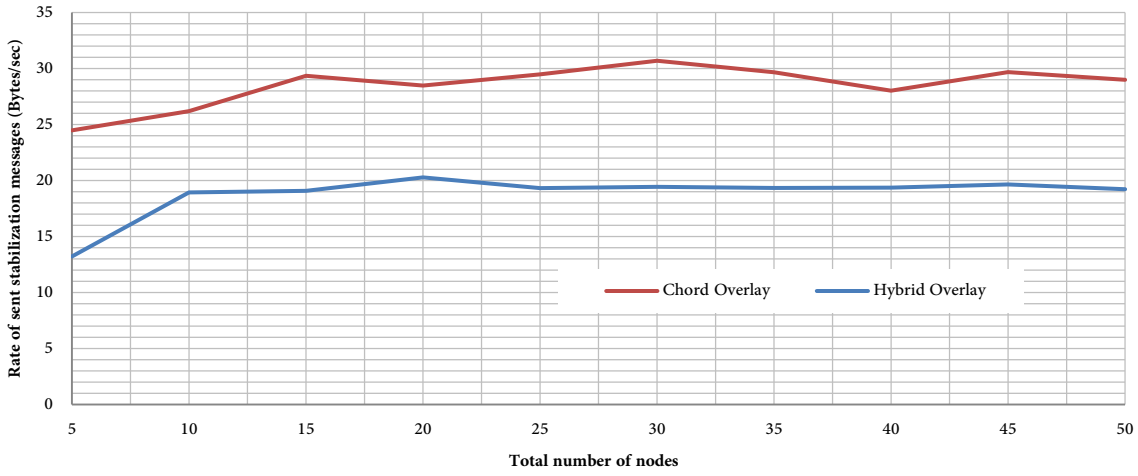


Figure 4.25 Mean rate of stabilization messages sent in Chord vs. H-SON.

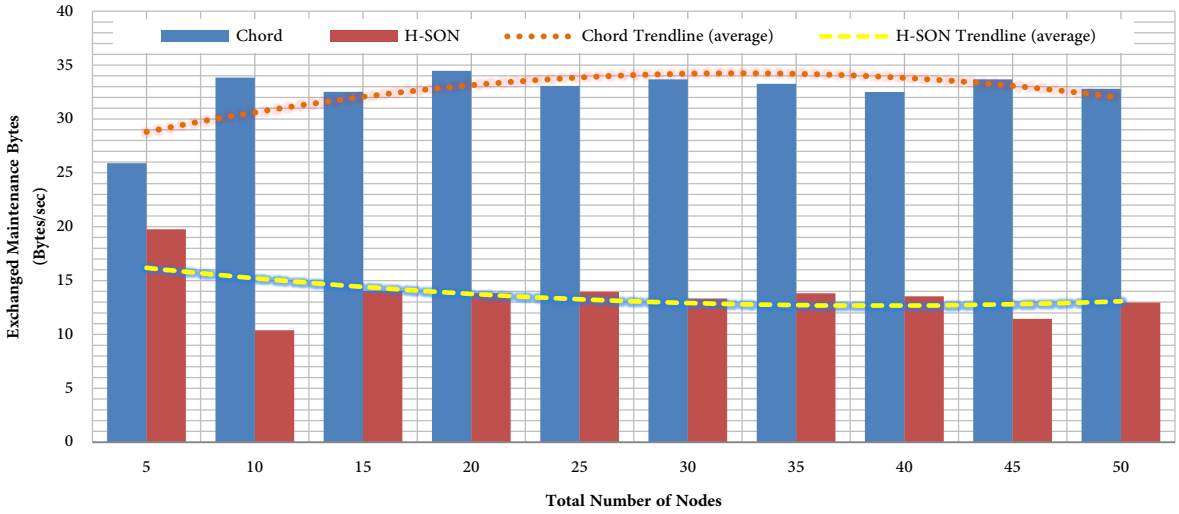


Figure 4.26 Total overlay maintenance bytes exchanged in Chord and H-SON.

A related measurement to that shown in Figure 4.26, is the mean rate of maintenance message bytes received by each node in the network to maintain existing links and fix broken ones. Figure 4.26 illustrates the expected high rate of messages received by each Chord node while the hybrid structure indicates a reduction of approximately 62% in the arrival rate of maintenance messages. Given the fact that Minor (leaf) nodes only receive messages from their Major (parent) nodes, the low rate associated with our hybrid overlay is understandable.

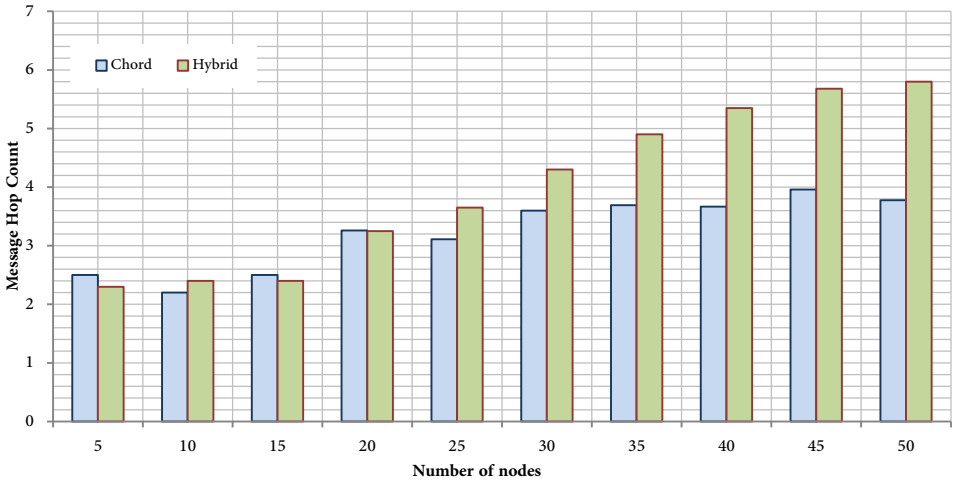


Figure 4.27 Average message hop count, Chord vs. H-SON.

One tradeoff noticed with the hybrid overlay is the expected number of hops a message must travel to reach its destination. Figure 4.27 illustrates the presence of a slight increase in the number of hops in the hybrid overlay over the Chord network. This is expected, because with the presence of a hybrid network, a worst case scenario arises when the source and the destination are both *minor nodes* at opposite ends of the overlay located in lower-level Chord structures. Such a limitation does not exist in a Chord network; however the benefits illustrated above of the proposed hybrid overlay outweigh this slight increase in hop count.

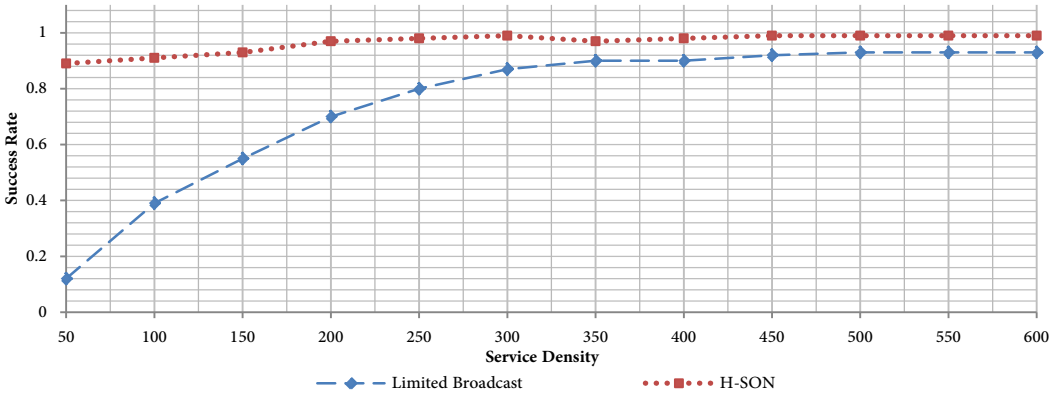


Figure 4.28 Service query success rate of H-SON vs. limited broadcast.

Query success rates measure the number of requests that receive positive responses by the total number of queries. Results in Figure 4.28 illustrate that our H-SON results in a higher success rate than limited broadcast (LB). Limited broadcast in our simulations represents a controlled broadcasting service query method that is confined by two variables: scope and hop count (depth). Scope refers to the number of direct neighboring MPs to which the current service node forwards its queries. Depth refers to the number of times a query is forwarded before being dropped. Our use of LB stems from the need to simulate a service discovery method that does not flood the network uncontrollably with unwanted queries.

The increase in success rate is visible in networks of both low and high numbers of services. Failures in our system occur only when the required service does not exist in the network, or when the MP providing the service does not have the required resources to support SSON composition.

4.7. Summary

This chapter presented our Hybird Service Overlay Network (H-SON). It represents a dynamic underlay encompassing all MPs, MSs, and MCs found in the network. Given the mobility of these entities, the H-SON displayed its adaptability to unexpected node movements and failures. Through sectorial divisions of the H-SON, any searches for specific types of service forming a composition can be done by searching within specified regions in the MP space. This region represents the total space within which needed types of service may exist regardless of the required QoS levels (stability, etc…) to form the composition or meet the MC's QoS requirements. To further reduce the physical search area for potential component services we have indicated that the dynamic promotion and demotion of overlay nodes in the lower overlay network results in movement of nodes with higher scores towards the center of the network. Therefore, nodes with higher stability levels, QoS, available resources, etc. are generally located closer to the center of the H-SON. Evaluating the distance away from the H-SON's center where potential service nodes can be located requires a non-exact evaluation method that reflects the non-exact nature of QoS properties' values. Consequently, a fuzzy logic based query system can be utilized to determine the band area within a H-SON sector where services of acceptable QoS levels are located.

Chapter 5

Plan-Based Service Composition

This chapter shows how the service composition problem as provided by the MC can be viewed by MSs as a planning problem. We present a unified methodology for representing MC requests, translating these requests into unique service composition plans, and enforcing generated plans by subsequent MPs until a completed path is formed. Section 5.1 discusses ontology-based modeling solutions that clearly define the roles and goals of different entities. Section 5.2 illustrates the ontology-based syntax we use for modeling services. Section 5.3 presents a model for structuring MC requests for SSON compositions. Section 5.4 presents a modeling approach for forming SSON composition plans. Finally Section 5.5 gives a general overview and concludes the chapter.

5.1. Ontology-Based Modeling

Service composition solutions concerned with providing autonomic, QoS-based services must adhere to; the requirements of self-management cycle of monitoring, analyzing, planning, and executing actions. Thus to enable automatic discovery and invocation of services without any human intervention, a formal definition of media requests and services is required. Having human interaction in the loop related to various service tasks such as discovery and composition, limits the scalability of service-oriented solutions and greatly curtails the composition performance. A formal description of services is needed to support interoperability between the diverse MCs, MPs, and MSs existing within the network. Here we provide a formal description of client's media requests, service composition plans

formulated by MSs, and service descriptions.

The reliance on the syntactic and semantic representation of operations (i.e. services) and the inputs and outputs of such operations raises the requirement for use of a unified modeling method. The presence of multiple entities in the proposed system along with the system's requirement for a common understanding of how concepts are represented and how they interact with each other, are all factors that encourage the use of a simple, sharable, and extendable approach for representing knowledge in a mobile environment. Consequently we have based our system on the assumption of the existence of a global ontology that unifies the view and understanding of any concept in the proposed approach.

Ontologies are used to describe concepts within an existing domain, as well as the relationships that may exist between these concepts. Ontologies are also used to share domain information knowledge between different entities and to simplify the ability to reason and deduce new context based on existing contexts and relationships. They could easily be projected into data structures for use by computers.

A number of markup languages have been used for the purpose of context modeling. The models used by researchers have either adopted some of these languages or tended to form their own methods for representing their acquired context. We have adopted the use of the Web Ontology Language (OWL) [113] [114] to model all ontologies within our presented system. OWL describes modeled data by using sets of classes and properties, and provides the flexibility and extensibility necessary within pervasive environments.

Within our architecture, we have assumed the presence of an OWL-based ontology covering an extended area of knowledge. This includes representations for services, and thus all components associated with such services. Additionally, the ontological representation extends to media requests formed by MCs, as well as service composition plans formulated by MSs to deliver media to the appropriate MCs according to the expected quality levels. Context is the information used to describe the situation of entities present within an environment whether these entities are human users, physical objects, or logical objects. Context is any information that can be sensed, inferred, or measured and might be of interest to a context consumer within a context-aware system. As such, context information is infinite in its most general definition. However, for the sake of simplicity,

each system usually defines context according to the requirements of its own domains. As such, context information within our proposed system is restricted to bandwidth, packet loss rates, available memory, and other information typically used within media service-oriented systems.

We have presented in past work a multi-layered ontology model that can further be utilized in our proposed service composition method [115] [116] [117] [118]. However, we are only interested in two out of the four ontology layers presented; Global Skeletal Ontology (GSO), and the Global Concrete Ontology (GCO).

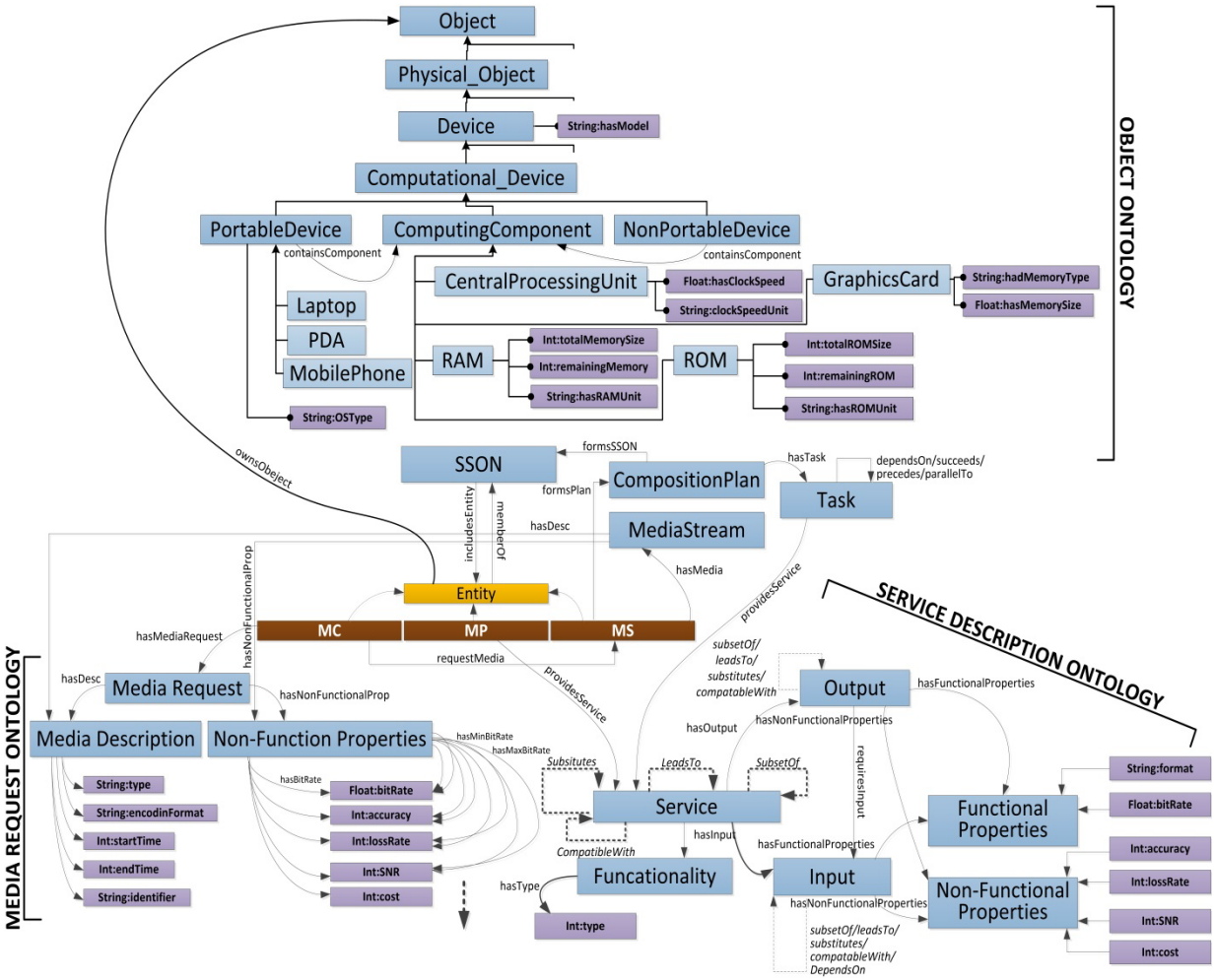


Figure 5.1 Simplified view of a global skeletal ontology for service composition.

The GSO, represents the base onto which all information within the system should be modeled. As seen in COMANTO [119] and SOCAM [120], ontologies were divided into an abstract high-level ontology and a lower domain-specific ontology. Using this concept of domain separation, our Global Skeletal Ontology can be seen as our modified version of the Upper or High-Level Ontology presented in both of these works. The approach adopted by Tao Gu et al. [121] and Strimpakou et al. [122] in their high-level ontologies was to create the most abstract set of classes at the high-level ontology and to leave the remaining details for the domain-specific ontologies. While creating the upper ontology is an easy task, however, designing domain-specific ontologies becomes more cumbersome, since many of the missing concepts and relationships must be created from scratch.

Through these ontologies, variations in the semantics of services and the syntactic representations of their functionalities can be compared. These comparisons can thus largely aid in searching for services that provide the exact functionality required in a composition plan designed by the MS as well as provide the quality levels and criteria that can meet the MC's requirements. Much work has been spent in the past modeling web services through ontologies [101][123][124][125][126][127]. These models can also be imported into, extended, and linked to our ontology to provide a complete view of service-based systems. As such, we only present here a partial view of what a GSO of service-based systems may include. This model contains the following components:

- The *entity roles* involved in the composition of service-specific overlay networks. These are organized into the following three distinctive roles introduced earlier:
    - *MC*: The entity interested in acquiring media from a MS.
    - *MS*: The entity providing some media to incoming MCs and responsible for forming a service composition plan to meet the MCs' requirements.
    - *MP*: a service-providing entity that can provide one or more services which could be utilized in a service composition path.
- The *entity functionality* or *service* model. Although this applies to each of the entity roles above, however for each role the associated functionality models differ. For MSs, the model describes the media provided along with any QoS properties in

addition to modeling any further services provided. For MPs, modeling is done for the services provided, and the inter-service relationships that may exist. In general, this describes the inputs, outputs, functions performed and their respective properties. Furthermore, this model describes the compatibility of services when interconnected to each other.

A general view of a possible GSO is illustrated in Figure 5.1.

5.2. Service Model

As the number of available services deployed increases, more and more services provide the same functionalities. We thus must separate the services types from service instances according to their different functionalities and quality levels. The service ontology model is a formal description of the service's functionality, and the interaction method with the service. The former describes its capabilities while the latter is a description of its interface. Both descriptions are further associated with a description of their acquired and provided quality levels. A general service model employed in our system is visualized in Figure 5.2.

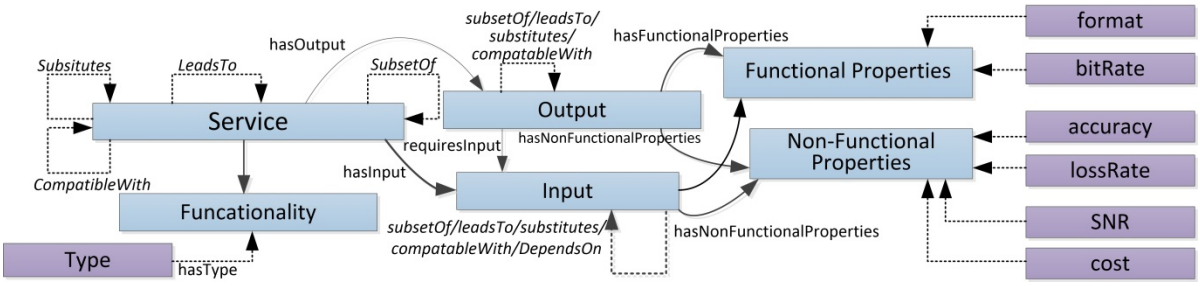


Figure 5.2 General ontology modeling of services.

Figure 5.2 illustrates that each service is divided into three main components; functionality, output and input. Functionality provides a description of the service at hand such as synchronization, media conversion, media joiner, media splitter, and so on. It is a description of the goals or objectives that are achieved by invoking the service. Inputs and outputs vary according to the service(s) provided. Inputs and outputs are described through

two property classes; functional and non-functional properties. The former includes the protocols and encoding formats that can describe the input or output of a service. The latter is a description of the qualities required for inputs and produced for outputs of the service. Consequently the QoS of any service in a composition is simply the non-functional properties of its output(s). The preconditions and consequences of invoking a service can thus be mapped directly to the input and output of such service.

Utilizing the above general ontological description of services, we have generalized service descriptions in our system as follows:

$$S_i = \{Op_i, In_i(in_{i,j}^{Description}, in_{i,k}^{QoS}), Out_i(out_{i,l}^{Description}, out_{i,m}^{QoS}), C_i, Dep_i\} \quad (5.1)$$

Op_i specifies the functions carried out by the service such as encoding, this is simply the functionality illustrated in Figure 5.2; In_i defines the acceptable input, this is divided into two specifications; input description $in_{i,j}^{Description}$ and input QoS $in_{i,j}^{QoS}$, where $1 \leq i \leq I$ refers to the service number, $1 \leq j \leq J$ is the set of input port stream functional properties (ex. encoding, bitrate, etc), and $1 \leq k \leq K$ is the set of input port QoS properties (ex. loss rate, accuracy, etc…). Similarly, $1 \leq l \leq L$ is the set of output port stream functional properties, and $1 \leq m \leq M$ is the set of output port QoS properties. The former for example specifies *{encoding, size, stream length,…}* of a video stream. The latter specifies the level of acceptable QoS, for example *{bit rate, loss rate, accuracy, …}*. This is the minimum acceptable input quality level required for the service to perform its functionality successfully and to guarantee the output QoS it specifies. Since MPs can be joiners/splitters, $j \geq 1$ is employed for each input/output port of the MP. As such, functional and non-functional descriptions are associated with every iteration of multi-input and multi-output of a service.

Out_i defines the characteristics of the output stream generated from the service. It has the same syntactic representation as In_i. C_i is an optional property representing an acceptable total cost for the MP's service. When formulating a service composition path from the MS to the MC, the total cost of invoked services has to be taken into consideration. Users are interested in maintaining costs under control and thus have a maximum level of acceptable prices they are willing to pay for acquired media and their associated component services. Finally, Dep_i is used to syntactically represent the dependency relation between

the MP service and preceding service(s). This is needed for services requiring two or more input streams from two or more different sources. An example is a media joiner that joins an audio stream and synchronizes it with a video stream. As such, this dependency must be modeled.

5.3. Modeling Media Requests

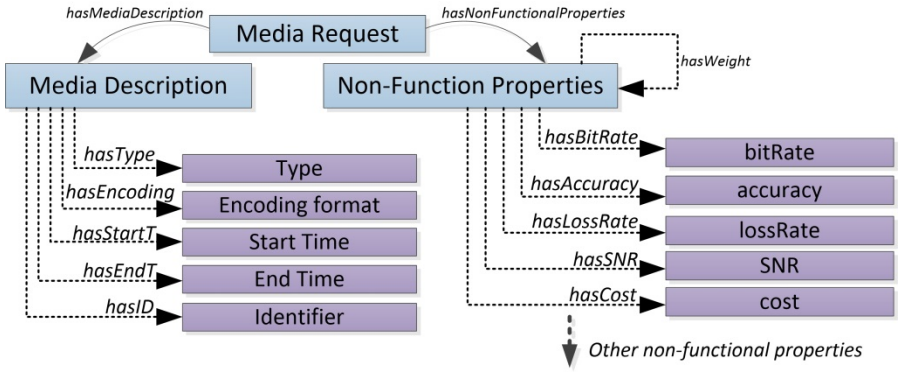


Figure 5.3 General ontology for MC media requests.

In a service-composition-oriented environment akin to that confronted here, a MC's media/service request R_i^M is presented as a composite of media description D_i^M parameter, and the required QoS properties' levels Q_i^M. M is an indicator the request type, media stream in this case, and *i* refers to the request number given that MCs can request more than one service at a time. The ontology model of MC media requests is provided in Figure 5.3. The figure visualizes the division of the request into the two criteria above. Each of these classes is further detailed with a set of properties that describe the exact interest of the MC. As such, the media request sent by the MC to a MS follows the following description:

$$R_i^M = (D_i^M, Q_i^M), \text{ where} \tag{5.2}$$

$$D_i^M = \{D_i^{type}, D_i^{id}, D_i^{encoding}, \ldots\} \text{ and} \tag{5.3}$$

$$Q_i^M = \begin{Bmatrix} Q_i^{delay}[P^{delay}, q_{min}^{delay}, q_{max}^{delay}], \\ Q_i^{jitter}[P^{jitter}, q_{min}^{jitter}, q_{max}^{jitter}], \\ Q_i^{cost}[P^{cost}, q_{min}^{cost}, q_{max}^{cost}], \\ \dots \end{Bmatrix} \quad (5.4)$$

The description parameter D_i^M encompasses general media description properties such as the media stream identity D_i^{id}, type D_i^{type}, encoding format $D_i^{encoding}$, etc., while the QoS parameter presents the MS with the MC's accepted levels of quality. Each QoS property $Q_i^x, where\ x = \{delay,\ jitter, cost, \dots\}$ is further expanded to define the acceptable range of values if not exact. These values range from a minimum acceptable value q_{min}^x to a maximum acceptable value q_{min}^x if present. Additionally, a priority level $P^x \in (1,10)$ is associated with each QoS property. The priority level illustrates the significance of the property to the MC. Thus, it would be more acceptable to provide a composition path that is less stringent regarding low priority QoS properties.

The syntactic and semantic properties of the MC's request are evaluated by the MS. The evaluation is done through a fuzzy inference engine further discussed in Chapter 6. An acceptable MC request shifts the MS to the planning phase. During this phase, the MS formulates a possible composition path plan $Plan_{MS}^{MC}$originating from itself towards the MC passing through n MPs, where $n \geq 0$. If $n = 0$ the media can be delivered directly to the MC, otherwise a service composition path has to be constructed.

5.4. Modeling Service Composition Plans

Media requests arriving from MCs to MSs may require the formation of service composition paths radiating from the MS towards the MC. The service composition problem can be viewed as a planning problem. The use of plans provides a solution to automatically compose services through mapping requirements into tasks descriptions that are potentially mapped into concrete services provided by MPs. Various previous works have relied on plans in composing services [128] [129] [130] [131] [132] [133].

A plan represents a full map of services required in every hop towards the MC along with the required levels of QoS. Planning can be defined as a problem solving technique

where a sequence of actions is derived from knowledge about available actions and their consequences [134]. The use of plans has been widely utilized in a variety of applications such as robotics, process planning and web-based information gathering. Given that services in our work form atomic components and given the need to enable dynamic and adaptive integration of these components into complex compositions, we thus need a methodology that can represent through a high-level abstract description the steps required to provide the completed composed service. Most work focusing on AI planning has relied on the formal description of service capabilities using ontologies; hence, the service model illustrated in the previous section. These techniques performed reasoning about their compositions using goal-oriented techniques. The user's original request is translated to model a sequence of services that can meet the presented goal.

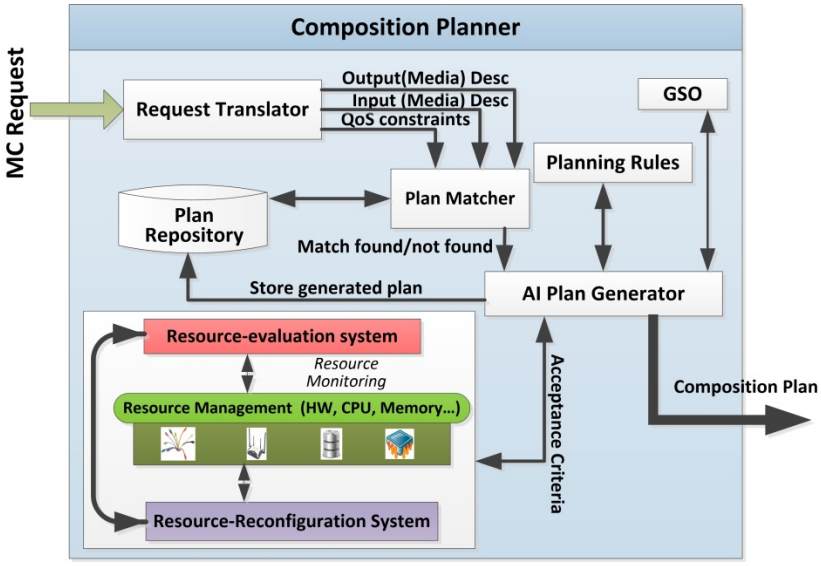


Figure 5.4 Architecture for service composition planner at the Media Server.

In our situation, a goal can be defined as the MC's original request in terms of delivered media, and its functional and non-functional descriptions. This reliance on AI planning has been proven to be well suited for autonomic environments where control for enforcing plans has to be distributed.

The composition can be viewed in the architecture shown in Figure 5.4. With the arrival of a MC's request at a MS, the request is broken down by a *Request Translator* into its three main components: input media description (ie the requested media identifier), the output

media description (functional properties), and the output QoS (non-functional properties). A *plan matcher* queries a *plan repository* to find a possible match regarding incoming requests. If a plan is found then the *Plan Generator* formalizes the plan and queries its own internal available resources to determine its capabilities of enforcing this particular plan. Every MS and MP provides one or more services. Each of these services is provided at certain QoS levels and requires a percentage of the node's resources. Consequently, prior to sending its acceptance to join an SSON composition following the retrieved plan, the current level of available resources must be compared to those required. If an acceptable level of resources are available; these resources must be reserved to prevent any conflicts with SSON compositions in the future. The plan generator relies on two components: the *GSO* and a set of *planning rules*. The former provides a representation for global concepts as discussed earlier and onto which all modeling in the system is based. The latter represents a set of rules entered by the administration that guide the MS in generating new SSON composition plans. The purpose of our work is not to present the details of how plans are formed and conceived. These details have been widely researched in the past [135] [136]. Once a new plan is generated it is stored within a repository for future use. The plan is passed on from the MS to subsequent MPs to form a composition path as will be presented in the next chapter.

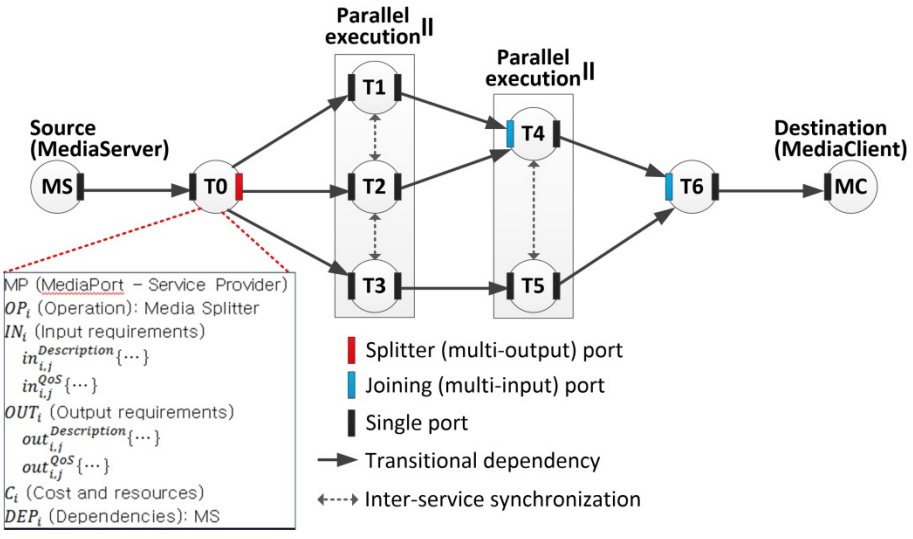


Figure 5.5 Sample SSON composition plan generated by the Media Server following the Media Client's requested specifications.

A graphical representation of an example composition plan is illustrated in Figure 5.5. A multistage composition path involving sequential and parallel services is presented. To facilitate chaining in service composition, MPs can be described according to their input and output ports. The concept was introduced in [99] [137]. *Single* input MPs take a media flow from one input port and transform it into a different output flow according to the service function they offer. Splitters have a single input and two or more output ports. A splitter might for example take an MP4 video stream as input, and produce an AVI video and MP3 audio streams as output. Joiners are the reverse of splitters, taking in two or more input streams and producing a single output stream.

As such, the general representation of a MS's composition plan as in Figure 5.5 is represented as follows:

$$Plan_{MS}^{MC} = \{MS, T_0,\ II(T_1, T_2, T_3),\ II(T_4, T_5), T_6, MC\} \tag{5.6}$$

where II represents services that must be executed in parallel, and where $T_i, 0 < i \leq n$ is a detailed description of the task that must be mapped to a required component service. For example, task *T4* is a joiner service that merges the audio stream of service *T1* with the video stream of *T2*. The parallel execution of tasks can have an effect on sequentially executed tasks even if it is not connected to its predecessor's parallel neighbors. For example, *T5* cannot execute without *T3*. However *T3*'s parallel execution with *T1* and *T2* must be synchronized as does *T5*'s execution with *T4*. This is due to the fact that *T6*'s final execution is affected by all previous services. Although complete and exact parallelism cannot be guaranteed, however we make the assumption that services executed in parallel synchronize their operations as visually illustrated in Figure 5.5. This synchronization can assure us that tasks execute without having any conflicts with parallel or sequential tasks.

Each task in a composition plan is modeled following (5.7), and discovered MP services must match such task descriptions to be considered are viable SSON composition candidates.

$$T_i = \begin{Bmatrix} Op_i, In_i\left(in_{i,j}^{Description}, in_{i,j}^{QoS}\right), \\ Out_i\left(out_{i,j}^{Description}, out_{i,j}^{QoS}\right), \\ C_i, \\ Dep_i \\ Synch(T_i, T_{i+1}, \dots) \end{Bmatrix} \tag{5.7}$$

Where Op_i refers to the service operation performed by the MP such as synchronization, buffering, etc... The formed plan represents an abstract map that has to be filled with concrete services found within our Hybrid Service Overlay Network. The abstract representation of *T0*'s Op_0 states that there must be a splitter service that acquires a media stream input with the description stated by a set of $in_i^{Description}, i \geq 1$ input descriptions and meeting the quality requirements stated in $in_{i,j}^{QoS}, where\ i,j \geq 1$. Given that each functional description $in_i^{Description}$ can have its unique set of non-functional characteristics, therefore each functional description is associated with a set of QoS properties

$$\left(in_{0,0}^{QoS}, in_{0,1}^{QoS}, in_{0,2}^{QoS}, \dots, in_{0,n}^{QoS}\right), \left(in_{1,0}^{QoS}, in_{1,1}^{QoS}, in_{1,2}^{QoS}, \dots, in_{1,m}^{QoS}\right), \dots \quad (5.8)$$

The same requirements apply to the output port description Out_i. Given that MCs can have a cutoff level of acceptable service prices, each task description must include a maximum acceptable cost, C_i, depending on the importance of the provided service. Any dependencies to previous services must also be declared through Dep_i. Finally parallel execution synchronization, if needed, must be clearly indicated in the $Synch()$ function.

The duty of the MS is to enforce the first step of the composition plan. This is done by performing a search for a suitable MP with a service description that matches or exceeds the requirements. Querying the MP space for the next MP in the plan path is based on a fuzzy-enhanced QoS-based search process. Details of our MP search algorithm are discussed further in Chapter 6. The search process returns a number of candidate MPs from which the best fitting one is chosen. Once this step is completed, the generated composition plan is passed from the MS to the next hop MP, then to the next, until the MC is reached and a valid composition path is established.

The cost of forming composition plans increases with the increased complexity of the MC's media stream requirements. We attempt in our future work to test the accuracy and cost of several plan formation strategies and the possibility of incorporating them into our work [131] [138] [139] [140]. Given the complexity of the planning process and the widespread variations in incoming media requests from MCs, simulations presented in this dissertation beginning with Chapter 7 onwards have restricted MC requests to a preconceived set of composition plans. The MS determines the best-fitting plan for the

SSON composition from which our presented service discovery and composition process follows.

5.5. Summary

The majority of existing dynamic composition techniques have focused on the semantics of offered services, their QoS, and automated composition planning. The organization of the H-SON detaches the need for a central coordinator entity to locate and combine services. Sectorial and band divisions along with dynamic promotion/demotion of nodes enable for a fast and efficient MP search method in compositions. This begins with a composition plan generated by the MS listing all tasks to be performed both sequentially and in parallel by the composed service. The plan represents a general guideline used by every component Media Port on the SSON path to confine its next-hop Media Port search. The initial composition plan generated by the MS from the MC request is utilized here to form a set of rules that guide future SSON compositions. Descriptions of required services' input and output ports as well as services' functionalities are clearly presented in each plan. At each hop the qualitative conditions (QoS) of the candidate next-hop can be modified according to the current node's QoS and the aggregated QoS of the composed path. Furthermore, modifications performed on established plans cannot at any moment violate the MC's requirements.

Chapter 6

Fuzzy Logic-Enhanced MP Search

Media delivery to MCs must abide to all requirements presented during the initial request stage. The quality of the delivered content should not conflict with the client's device capabilities, nor should it contradict the QoS requirements specifically listed by the client in its request. However, when dealing with QoS, exact values are difficult to obtain. The acceptable accuracy level of a media stream delivered by a MS, or tolerable jitter levels are subjective properties. Values of non-functional properties vary dynamically depending on available resources at the MP and the load introduced by currently-running services of SSONs with which a MP is currently associated. Consequently, searching for MPs based on QoS must employ a mode of reasoning that is approximate rather than exact. We have thus chosen to use fuzzy logic (FL) to enhance our MP search method providing an approximation approach to choosing appropriate MPs in SSON compositions.

The remainder of this chapter is organized as follows. Section 6.1 provides an overview of Fuzzy Logic and discusses the benefits foreseen in its use in service composition. Section 6.2 discusses the FL QoS model chosen in our system. It provides a detailed description on how non-functional properties of a service are modeled, aggregated, and an overall evaluation of the service's QoS is retrieved. Section 6.3 discusses the MP search process that combines the FL model and our H-SON to determine the MP sectors and bands where appropriate MPs can be found during the composition phase. Finally, Section 6.4 provides a summary and discussion of the main concepts discussed in this chapter.

6.1. Overview of Fuzzy Logic

Fuzzy logic (FL) was introduced as an approach to mimic the human mind in reasoning on concepts considered to be approximate rather than exact. In traditional computing and decision-making, actions were based on precisions and certainty. However, imprecise values and concepts require a greater degree of tolerance for error and uncertainty. As such, FL enables us to specify mapping rules in terms of words rather than numbers. This approach increases our horizon of acceptable values for any concept. FL relies on simple *if-then* rules rather than mathematically modeling systems. The difference is its reliance on fuzzy antecedents and fuzzy consequents. Thus the system relies on the operator's experience rather than the technical understanding of the system. Consequently this approach is highly fit for QoS and QoE-based systems that rely highly on the users' experience of received media. Moreover, fuzzy systems can model nonlinear functions of arbitrary complexity to any desired degree of accuracy, as well as multi-input, multi-output systems.

Fuzzy Logic was introduced by Zadeh [141] to provide an opportunity to model conditions that are inherently imprecisely defined. The membership function of a fuzzy set has a range value in the interval [0, 1]. In the following we present some of the basic notions related to fuzzy theory related to our work. There are five features that render FL a particularly beneficial choice for use in designing a system. First is its inherent robustness since it does not require precise inputs. Moreover, FL produces smooth control outputs even with the presence of a wide range of input variations. Second, the reliance of the system on a set of defined *if-then* rules, future modifications can improve the system by simply altering these rule sets. Third, because of the rule-based operations of FL, any number of inputs can be processed and any number of outputs can be generated. However, with increasing number of inputs, the complexity of the rule set increases. As such, the problem can be easily broken down into smaller chunks resulting in a number of simple FL systems. Finally, the fourth benefit stems from FL's ability to evaluate nonlinear systems that are difficult or even impossible to model mathematically.

Fuzzy sets are an extension of crisp sets. The difference between the two stems from the fact that the latter allows full membership or no membership at all, while the former allows

for partial membership. Consequently, a fuzzy set A on a universe of discourse U is characterized by a membership function $u(x)$. The value of $u(x)$ ranges from 0 for complete exclusion from *A*, i.e.$x \notin A$, to complete inclusion, $u(x) = 1$. These fuzzy sets are used to represent linguistic variables such as slow, fast, far, near, heavy, etc. Each element can also be a member of more than one fuzzy set at a time. Ordered pairs can be used to represent a fuzzy set *A* in *U*. Each pair consists of a generic element *x* and the value of its membership function:

$$A = \{(x, \mu_A(x)) | x \in U\} \tag{6.1}$$

The linguistic variable *x* is characterized by

$$T(x) = \{T_x^1, T_x^2, \dots, T_x^k\}, \text{ and} \tag{6.2}$$

$$\mu(x) = \{\mu_x^1, \mu_x^2, \dots, \mu_x^k\} \tag{6.3}$$

where *T(x)* is the linguistic value set of *x* referred to as the *term set*. Each T_x^i is a fuzzy number with membership function μ_x^i defined on *U*. For example, if *x* indicates the loss rate of a service provided by MP_i, then T_x^i may refer to sets such as *high*, *acceptable*, or *low* loss rates. Each of these sets is associated with a universe of discourse with values that tend to overlap and form smooth transitions reflecting their uncertainties. Consequently a MP service can have a loss rate that is a member of the *high* and *acceptable* loss ranges simultaneously, but with varying degrees of membership.

Logical operations can be performed on fuzzy sets. The most elementary operations available are union (OR), intersection (AND), and complement (NOT). Given two sets *X* and *Y*, their union $X \cup Y$ contains all elements found in either *X* or *Y*. Consequently, if an element $\alpha \in X$ or $\alpha \in Y$ then $\mu_{X \cup Y}(\alpha) = 1$. The intersection of the two sets, $X \cap Y$, contains all elements that are members of both *X* and *Y*. Consequently, $\mu_{X \cap Y}(\alpha) = 1, if\ \alpha \in X\ and\ \alpha \in Y$. Finally, the complement points to all members that are not present within the indicated set. That is $\mu_{\bar{X}}(\alpha) = 1\ if\ \alpha \notin X$, otherwise $\mu_{\bar{X}}(\alpha) = 0$. However, given the fact that in FL the truth of any statement is a matter of degree, these three operations have to be mapped to operators that preserve their respective results. This is done through max, min and complement operations.

$$\mu_{X \cup Y}(\alpha) = \max[\mu_X(\alpha), \mu_Y(\alpha)] = \mu_X(\alpha) + \mu_Y(\alpha) - \mu_X(\alpha)\mu_Y(\alpha) \tag{6.4}$$

$$\mu_{X\cap Y}(\alpha) = \min[\mu_X(\alpha), \mu_Y(\alpha)] = \mu_X(\alpha)\mu_Y(\alpha) = T\big(\mu_X(\alpha), \mu_Y(\alpha)\big) \tag{6.5}$$

where *T* is a binary mapping that aggregates the two membership functions.

$$\mu_{\bar{X}}(\alpha) = 1 - \mu_X(\alpha) \tag{6.6}$$

The above *T* binary mapping is referred to as T-norm, or triangular norm. Similarly, fuzzy union operations in (6.4) can be specified by an S-norm mapping operation as follows: $\mu_{X\cup Y}(\alpha) = S\big(\mu_X(\alpha), \mu_Y(\alpha)\big)$. However, more applications use min and max for fuzzy intersection and union respectively.

As stated above, FL rules assume the form of "*if* α *is A, then* β *is B*", where $\alpha \in U$ and $\beta \in V$, *U* and *V* are two universes of discourse, and $\mu_{A\to B}(\alpha, \beta) \in [0,1]$. Evaluating such rules requires fuzzifying the input and applying the necessary fuzzy operators. Then implication is performed by applying the result of the antecedent to the consequent thus evaluating the membership function $\mu_{A\to B}(\alpha, \beta)$. In FL, a rule is fired as long as there is a nonzero degree of similarity between the premise and the antecedent of the rule.

Any system relying on FL requires a fuzzy inference system (FIS). An FIS defines a nonlinear mapping of the input data vector into a scalar output through fuzzy rules. This process involves the input and output membership functions, FL operators, fuzzy rules, aggregation of the output sets, and a defuzzification process. This process is illustrated in Figure 6.1 redrawn from [142].

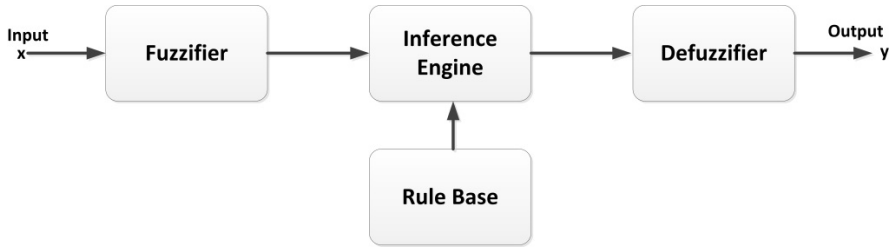


Figure 6.1 Fuzzy Logic (FL) system (redrawn from [142]).

The four main components of any FL system are:

1. *Fuzzifier*: Acquires crisp input numbers and maps them into the corresponding fuzzy functions. This is performed by taking the input values and determining the

degree to which they belong to each of the fuzzy sets through membership functions. The fuzzifier's job is required to activate all the rules that are written in terms of linguistic variables.

2. *Inference Engine*: Used to map the input fuzzy sets into the output fuzzy sets. The inference engine determines, for each rule, the degree to which the antecedent is satisfied. Rule antecedents composed of more than one clause require an aggregation process that results in a single number representing the output of the antecedent for each of these rules. In the event such aggregation is required, the output fuzzy sets of all involved rules need to also be combined into a single fuzzy set. The order in which rules are fired in FL is insignificant.

3. *Defuzzifier*: Maps the output fuzzy sets into crisp values. There are a number of defuzzificaton methods such as centroid, maximum, mean of maxima, height and modified height. In our proposed solution we utilize the centroid method which calculates the center of gravity of the aggregated fuzzy set. Our use of the centroid method stems from its wide use and its production of an output that tends to move smoothly around the output fuzzy region.

4. *Rule Base*: Contains a set of rules entered by system administration. In our proposed solution, each MS must have a rule repository which could be used to form SSON composition plans, and evaluate the validity of service quality levels. Furthermore, given the autonomic nature of our system's environment, dynamic modifications to these rule bases are also possible.

The interpretation of fuzzy rules is done in three stages:

1. Acquire a (0 to 1) degree of membership by resolving the fuzzy statements in the antecedent.

2. In the case of multiple parts to the antecedent, fuzzy logic operators are applied to resolve the antecedent to a single number between 0 and 1. This value represents the degree of support for the rule. Consequently, if we consider the *ith* rule of a multi-input rule $R_i: if\ x_1\ is\ T_1^i\ and\ x_2\ is\ T_2^i\ then\ y\ is\ T_y^i$ then the firing strength of the rule can be defined as $\gamma_i = \min\left(\mu_{x_1}^i(x_1),\ \mu_{x_2}^i(x_2)\right)$ or $\gamma_i = \mu_{x_1}^i(x_1)\mu_{x_2}^i(x_2)$.

3. Shape the output fuzzy set by applying the implication method. This is done by using the degree of support for the entire rule. The implication process takes a single number acquired by the antecedent and produces an output fuzzy set using the minimum and product methods. These are represented as $\mu_y^i(w)' = \min\left(\gamma_i,\ \mu_y^i(w)\right)$ and $\delta_y^i(w)' = \gamma_i \mu_y^i(w)$ respectively, where *w* is the support value of the membership function.

Once the firing strengths of the rules involved in the fuzzy system have been found, the output fuzzy sets must be combined into a single composite fuzzy set. This is referred to as aggregation, and it takes all the fuzzy sets of the outputs of fired rules and combines them into a single input that is used in the defuzzifier to get a single crisp value. One of the most commonly used aggregation methods is the max method:

$$\mu_y(w) = \max\left(\mu_y^1(w)\mu_y^2(w)\right) \tag{6.7}$$

where and are two fuzzy sets representing the outputs of two fired rules.

As stated earlier, we use the centroid defuzzification method in our fuzzy-enhanced MP search process in the proposed H-SON. This method determines the center of gravity; y_i', of the fuzzy output set B. For continuous aggregated fuzzy set, the centroid is given by:

$$y' = \frac{\int_S\ y_i \mu_B(y) dy}{\int_S\ \mu_B(y) dy} \tag{6.8}$$

where S denotes the support of $\mu_B(y)$. A simpler approach to the centroid method can be used as an approximation which uses summation as follow:

$$y' = \frac{\sum_{i=1}^{n} y_i \mu_B(y_i)}{\sum_{i=1}^{n} \mu_B(y_i)} \tag{6.9}$$

This section provided a quick overview of FL and how it can be employed as a reasoning method for values that are approximations rather than exact. Section 6.2 discusses how we use a Fuzzy Logic System to evaluate QoS and aid in searching for MPs in our H-SON.

6.2. Establishing a Fuzzy QoS Model

Section 6.1 introduced the presence of various membership function shapes that can be

used in FL including triangular, trapezoidal, generalized bell shaped, Gaussian curves, polynomial curves, and sigmoid functions. Each of these functions has its benefits and the type and number of parameters used in each differs. QoS properties used to describe different quality aspects of services include response time, latency, availability, accessibility, security, accuracy, etc. Some QoS properties can be expanded to include several sub-properties for different evaluation criteria. For example, the efficiency of a service can be expanded to include latency, response time, availability, and so on. Furthermore, QoS properties may be evaluated by one or more metrics. These metrics can be unitless or measured by one or more units. Moreover, the importance of each QoS property (reflected in the weight factors discussed in Chapter 4 in the node score equation (4.2)) varies from one property to another. Consequently, an imprecise method such as that of fuzzy based models could be used to evaluate where a MP's service quality stands in comparison to other functionally-similar services.

A number of works have used fuzzy logic for service ranking and selection [143] [144] [145]. In [146], Chien et al. divided QoS criteria into five fuzzy sets that describe their constraint levels. These sets were Poorly Acceptable (PA), Almost Acceptable (AA), Acceptable (A), Very Acceptable (VA), and Extremely Acceptable (EA). The sets were used in a notion $FuzzyS(P_{WSDP}, q, l)$ where P_{WSDP} denotes the composition problem at hand, and l is the level according to the QoS criterion q. For all QoS criteria, the relationships between the five QoS satisfaction levels were given in five unique membership functions. The overall QoS preference of the user was represented by a fuzzy aggregation of the five fuzzy sets through the fuzzy AND logical operation. In [147], Xiong et al. modeled QoS-based service selection as a fuzzy multiple criteria decision making problem (*FMCDM*). Two sets of linguistic expressions were used; for QoS criteria and for the weights of the QoS criteria. A triangular fuzzy membership function was used to map the linguistic expressions. Selecting the best service was performed in three steps. First a fuzzy judgment matrix was constructed, normalized and defuzzified, with triangular fuzzy members for the m QoS criteria and n candidate web services. Second the linguistic expressions for QoS weights are defuzzified and normalized, the entropy weights are computed, and the synthetic weights of the QoS criteria are calculated through linear combination of the QoS weights and the entropy weights. Finally, the overall quality score

of each service is computed using the weighted sum method.

For the purpose of our work, we have employed the generalized bell-shaped membership function for its adjustability and ability to fit the behavior of QoS properties. The bell-shaped membership function is given by:

$$f(x;b,c) = \begin{cases} S\left(x; c-b, c-\frac{b}{2}, c\right) & for\ x \le c \\ 1 - S\left(x; c, c+\frac{b}{2}, c+b\right) & for\ x > c \end{cases} \tag{6.10}$$

where $S(x;a,b,c)$ represents a membership function defined as

$$S(x;a,b,c) = \begin{cases} 0 & for\ x < a \\ \frac{2(x-a)^2}{(c-a)^2} & for\ a \le x < b \\ 1 - \frac{2(x-c)^2}{(c-a)^2} & for\ b \le x \le c \\ 1 & for\ x > c \end{cases} \tag{6.11}$$

A generalized representation of the membership function is illustrated in Figure 6.2. Any desired generalized bell-shaped membership function can be obtained by the proper selection of the parameter set (a, b, c). By adjusting *c* and *a* we can vary the center point and the width of the membership function. Moreover, adjusting the value of *b* can be used to control the slopes at the crossover point. A crossover point refers to the point on a bell curve where the slope changes from positive to negative or vice versa.

Service selection is an important step in SSON composition. Our use of an H-SON permits us to determine the exact overlay sector bands within which potential MPs providing such services with the required QoS levels can be located. With the vast increase in the number of services available that are functionally identical, QoS plays a major role in differentiating these services. However, the difficulty of precisely defining values of QoS properties requires the use of FL. Similar to the way MPs deliver descriptions of their services, the quality of these services is also provided during the H-SON join process which determines the location of their placement in the H-SON. Taking the MC's received media quality constraints consideration, each MP falling within the determined sector band can internally evaluate its overall quality degree in comparison to the service request's requirements.

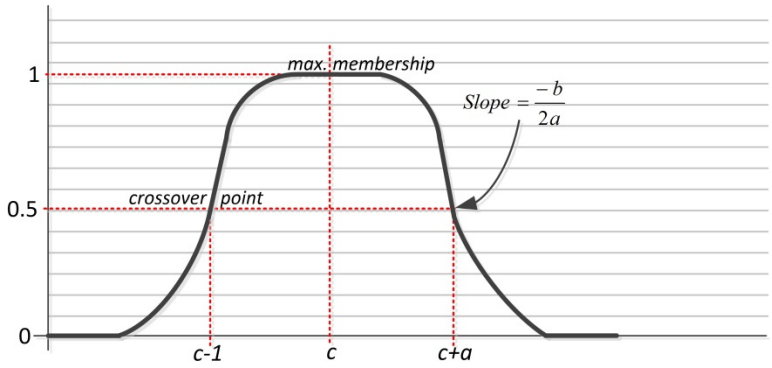


Figure 6.2 Fuzzy bell-shaped curve (redrawn from [142]).

Membership functions used in our systems begin and end with sigmoidal curves. The beginning curve exhibits a progression behavior from a climax: beginning, accelerating and approaching a small end. The end curve exhibits a reversed behavior that begins with a small value and progresses to a maximized end. The sigmoid function is modeled as follows:

$$f(x;a,c) = \frac{1}{1+\exp[-a(x-c)]} \tag{6.12}$$

where a controls the slope at the crossover point $x=c$. Depending on the sign of the parameter a, the sigmoid membership function is inherently open right or open left. Once again, steeper slopes represent QoS property values with high priority levels.

From (6.10) and (6.11), a, b, and c are parameters that are adjusted to fit the desired membership data. To reflect the dynamic behavior of QoS properties, we assume that c, located at the center of the membership curve, represents the exact desired value: $q^j_{required}$. Parameter a determines the width of the membership curve and represents the set's range values that are likely to fall within the specified class with substantial certainty: $q^j_{min} < a < q^j_{max}$. Parameter b reflects the slope of the curve. In our case, the slope reflects the extent of the value's accuracy to the MC's requested quality level. The steeper the slope the more sensitive the decision is to the value's accuracy level.

We exemplify the above discussion with a simple example. If we assume that a MC has requested to receive some type of media stream from a MS under some conditions, one of which was that the total loss rate of transferred packets (equation (5.4) Section 5.3) should

be around $Q_{loss}^{M} = 1.35\%$. Moreover, the MC's request may indicate that the maximum loss rate should not exceed $q_{max}^{delay} = 2\%$ at the worst case and can be as low as $q_{min}^{delay} = 0.7\%$ in the best case scenario. It is clear why an upper bound on loss rate is needed since increased loss rate reflect a large deterioration in QoS. For each QoS priority the reasoning behind stating upper and lower bounds is different. One possible reason for the example at hand could arise from the fact that media delivery at very low loss rate result in higher cost expectations. Consequently, knowing that most MPs providing services at such high quality levels are likely to exceed the MC's total composition budget, it is illogical to forward query messages in the network to MPs whose services are going to be rejected during the SSON composition phase. Furthermore, the MC's request includes a priority level for the loss rate on a scale 1 to 10; 1 being the highest, such as $P^{loss} = 3$.

The above information can be used to shape the bell-shaped fuzzy membership function. Given that our system is established in a mobile and autonomous environment, the details associated with how to model the fuzzy system depends on the following factors:

1. The entity (MS, or MP) that has received the composition request. Evaluating FL functions with a large number of linguistic expressions consumes a larger percentage of the entity's resources than one with a small number. Consequently it is an internal decision reflecting the entity's willingness to use its resources for such process.

2. The accuracy with which the QoS priority level should be evaluated. With more linguistic expressions in a membership function, the fuzzy evaluation increases in its accuracy.

3. The precision of the QoS property currently under consideration. This is directly related to the current stage of the composition path. Sometimes properties only require two linguistic expressions $L_i = \{accept, reject\}$ while at others more expressions are needed $L_i = \{strong\ accept,\ weak\ accept,\ weak\ reject,\ strong\ accept\}$. Under most circumstances as the composition path is formed, the window for precision flexibility becomes narrower as the path nears completion. The reasoning behind such observation arises from the fact that intermediate MPs usually offer their services at levels that are very close to the

bare minimum of the MC's requirements. Consequently, every MP service in the SSON path receiving an input stream must produce an output stream with a narrower range of acceptable QoS level.

A possible membership function for the above example is illustrated in Figure 6.3. The loss rate value range has been divided into three linguistic expressions: $L_{loss} = \{Low, Acceptable, High\}$. The midpoint c is set directly from the MC's required loss rate at 1.35%. The two end points of acceptable loss rates are set to 0.7 and 2.0 taken from q_{min}^{delay} and q_{max}^{delay} respectively. By determining the MC's restrictiveness to the exact value of the stream's loss rate (a subjective reasoning process), three adjustments could possibly be employed by the entity performing the fuzzy evaluation. First, the end points can be shifted as desired to increase the range of acceptable loss rates. Second, by adjusting the slope of the bell-shaped curve, the degree of membership of the input value can belong with higher degrees to one of the three linguistic expressions. Third, the diagonal shift of the linguistic expressions can increase or decrease their intersection areas. A fuzzy system with larger areas of intersection reflect the uncertainty of what constitutes a value belonging to one linguistic expression from another, and a less restrictive QoS property with lower priority level $1 \leq P^{loss} \leq 10$.

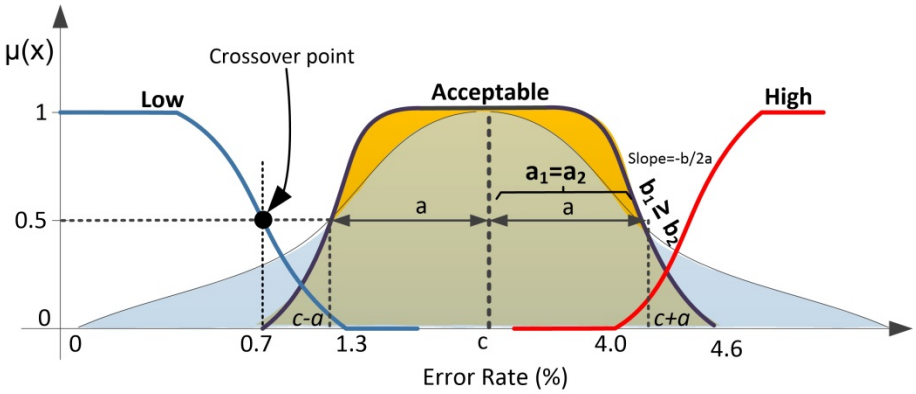


Figure 6.3 Sample usage of bell-shaped curve for fuzzy evaluation of media loss rate.

The next section discusses the steps required to form a composition path that combines our H-SON and the bell-shaped fuzzy membership functions. We illustrate how a composition plan is passed from one MP to another, each evaluating its internal QoS, and how a candidate is chosen from functionally-similar MPs.

6.3. Fuzzy H-SON MP Sector and Band Search

When a MC sends a request to a MS, a service composition plan is inaugurated. The MS performs an internal fuzzy-based evaluation of the requested media QoS in comparison to its provided media. Evaluating the fuzzy sectors and adjusting the respective *a*, *b*, and *c* variables in (6.10), (6.11) and (6.12), the MS must decide whether the request should be accepted. Acceptance is determined by finding the percentage of QoS properties whose values, as provided by the MS, fall within the 'acceptable' membership range. This must also take into consideration the priority level of each property, thus permitting some level of violation for low-priority properties. Figure 6.4 illustrates a partial example media request sent by a MC:

REQUEST: R_{MC}^{Media} sent to MS_{id}
[$\boldsymbol{D_{MC}^{Media}}$ [%description of the media being requested
D_{type}^{Media} = "video stream";
D_{id}^{Media} = "WALL-E";
$D_{encoding}^{Media}$ = "MP4";
] $\boldsymbol{Q_{MC}^{Media}}$ [%Quality of received media stream.
$\boldsymbol{Q_{delay}^{Media}}$ [
$q_{required}^{dely}$ = 80 msec;
q_{min}^{dely} = 50 msec;
q_{max}^{dely} = 160 msec;
P^{delay} = 4;
]$\boldsymbol{Q_{jitter}^{Media}}$ [%jitter or received media stream
$q_{required}^{jitter}$ = 0.8%;
q_{min}^{jitter} = 0%;
q_{max}^{jitter} = 1.3%;
P^{jitter} = 9;
] **...**
$\boldsymbol{Q_{cost}^{Media}}$ [%cost unimportant to MC, no limits set and has lowest priority
q_{min}^{dely} = $-;
q_{max}^{dely} =$ -;
P^{delay} = 1;]
]
]

Figure 6.4 Sample media request as sent by the Media Client to the Media Server.

A MS receiving the request determines whether it can provide the identified video stream. The evaluation is performed on two fronts; First, whether the video is available for delivery, and second if the QoS of the present video matches or exceeds the requested quality level. For the second front, the evaluation utilizes the discussed fuzzy logic system. This is done by establishing the necessary fuzzy membership functions for each of the QoS properties. An example is illustrated in Figure 6.5.

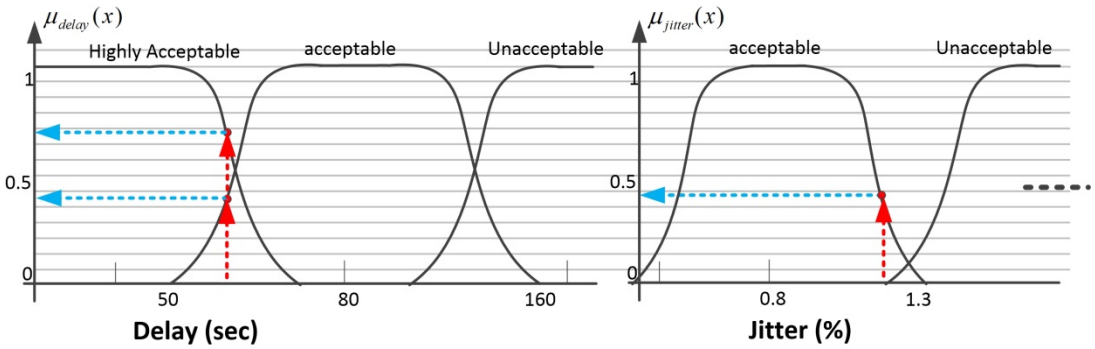


Figure 6.5 Possible fuzzy membership functions for two QoS properties (Delay, Jitter). Each presents varying membership parameter values and the MP's provided QoS level (red dotted line).

Only two sets of membership functions are illustrated in the example (delay and jitter). The low priority of delay permits the division of linguistic expressions into three categories (highly acceptable, acceptable, and unacceptable). Furthermore, it permits larger areas of intersection between the expressions. However, the high priority of jitter (value of 9, from Figure 6.4) results in a limitation of two linguistic expressions (acceptable and unacceptable). Furthermore, altering the bell-curve's variables (a, b, and c in (6.10) and (6.11)) shapes the acceptance region to shift closely towards a minimized jitter percentage. Additionally, the intersection of the acceptable and unacceptable regions has been kept to a minimum. The MS inputs its internal QoS capabilities into the fuzzy membership functions (ex. 68msec for delay and 1.1% for jitter). Depending on the rule set stored within the local rule depository (Figure 6.1) at the MS, the set of rules triggered and their respective firing strengths are determined. A single value representing the degree of acceptability, 0 to 1, of each QoS property is determined from the fuzzy logic system. The total acceptability is then evaluated according to (6.13)

$$Acceptability_{Total}^{QoS} = \left(\sum_{i=0}^{n-1} AcceptDeg_{QoS_{Prop_i}}\right)/n - (\sum_{i=0}^{n-1} RejectDeg_{QoS\_Prop_i})/n \quad (6.13)$$

where $-1 \leq Acceptability_{Total}^{QoS} \geq 1$, $AcceptDeg_{QoS\_Prop_i}$ represents the degree of acceptability of QoS property $i = \{delay, jitter, lossRate, \dots\}$, and $RejectDeg_{QoS\_Prop_i}$ represents the degree of rejection of the same property *i*. Total acceptability is generated from all *n* QoS properties. Both acceptance and rejection are included due to the possible intersection of the two linguistic expressions in the fuzzy membership function. Negative acceptability results in the automatic rejection of the MC's media request. Contrarily, positive acceptability with values close to 1 results in an acceptance. However, values $> 0\ and \ll 1$ fall within an uncertain region where it is up to the entity at hand to make a decision.

Each media and/or service provided by MSs and MPs in addition requires a certain amount of resources: $Cost_{resources}^{S_i} = \{resource_0, resource_1, \dots, resource_{n-1}\}$, where $resource_i = \{cpu, memeory, \dots\}\ for\ 0 \geq i < n$ resources. If the required resources to deliver a media stream or invoke the required service do not exceed available resources, i.e. $Resources_{available}^{MS_j} \geq Resources_{required}^{S_i}$, and the aforementioned QoS falls within the acceptance region, then the decision is to accept joining the SSON.

According to the composition plan, the process of composing an SSON continues by searching for the candidate next-hop MP that provides the required service; service *T0* in Figure 5.5. The search process relies on our H-SON. As such, the search process can be summarized in the following steps:

1. Find the MP sector based on the *type* of service required.
2. Evaluate the required QoS level for the current step in the composition through the fuzzy-logic system described above, and retrieve the sector bands' borders, (r_1^i, r_2^i), within which potential MPs are located. The composition query is forwarded to that region of the H-SON.
3. Each MP receiving the request performs an internal evaluation of its available local resources, types of services provided, and QoS levels. If the node deems itself capable of participating in the SSON composition, an acceptance message is sent

back to the previous node.

4. The sender ranks all acceptable nodes and chooses the best-fit MP for the next hop.

The above steps are repeated for each hop in the SSON path until the MC is reached. Step 1 utilizes a semantic and syntactic similarity evaluation of *T0* (from Figure 5.5). The evaluation result retrieves the sector area bounded by $(\alpha_{s_i}, \beta_{s_i})$. Details regarding semantic and syntactic similarity are discussed further in Chapter 7. The challenge in this step is related to how to perform a comparison and logical transformations on media endpoints of the services as well as the functionality of these services. Given the dynamicity of the H-SON, sector borders could as well change over time. Consequently, a request to the H-SON management layer is sent for an update to the general overlay organization. The reply illustrates the number of sectors *n*, the type of service provided in each sector $\{s_0^{type}, s_1^{type}, \dots, s_{n-1}^{type}\}$, and the borderline angles bounding the sectors $\{(\alpha_{s_0}, \beta_{s_0}), (\alpha_{s_1}, \beta_{s_1}), \dots, (\alpha_{s_{n-1}}, \beta_{s_{n-1}})\}$. A comparison between the next-hop service needed according to the plan and the available service sectors determines the MP sector to be searched.

The search area must then be reduced to locate potentially acceptable next-hop MPs. This is done by a fuzzification process of the required end-to-end QoS of the media stream.

To perform step 2 above, recall that each MP sector in our H-SON was broken down into QoS-based bands. The bands' low and high bound borders, (r_1^i, r_2^i), were mapped according to the minimum and maximum node scores members of the sector band respectively. Node scores depend on the MPs' QoS and their overall stability. The MS's output stream is delivered at a quality level $Q_{MS} = \{Q_{MS}^{delay}, Q_{MS}^{jitter}, Q_{MS}^{cost}, \dots\}$ that may equal, exceed, or fall below the MC's requested quality Q_i^M in (5.4). Accordingly, the acceptable level of QoS for the next-hop MP input and output ports must be modified. The modifications are performed to the MC request's QoS levels in (5.4). Modifications can be performed on all QoS variables stated in the MC's request. Such modifications directly alter the fuzzy logic system's shape as discussed in Section 6.2. The width, number, and steepness of membership sets are altered for each QoS property according to:

- Importance of the QoS property.

- Aggregated QoS level of the composition from the MS up to the previous SSON hop.
- Accepted QoS properties' values as established by the MC and the MS-generated composition plan.

Modified QoS properties' values are inserted into the modified fuzzy systems. Additionally, the expected composition lifetime is included in the fuzzy system. The resultant output fuzzy membership function determines the searchable band, (r_1^i, r_2^i), within the sector. Each band is divided according to a node scoring system we introduced earlier in Chapter 4.

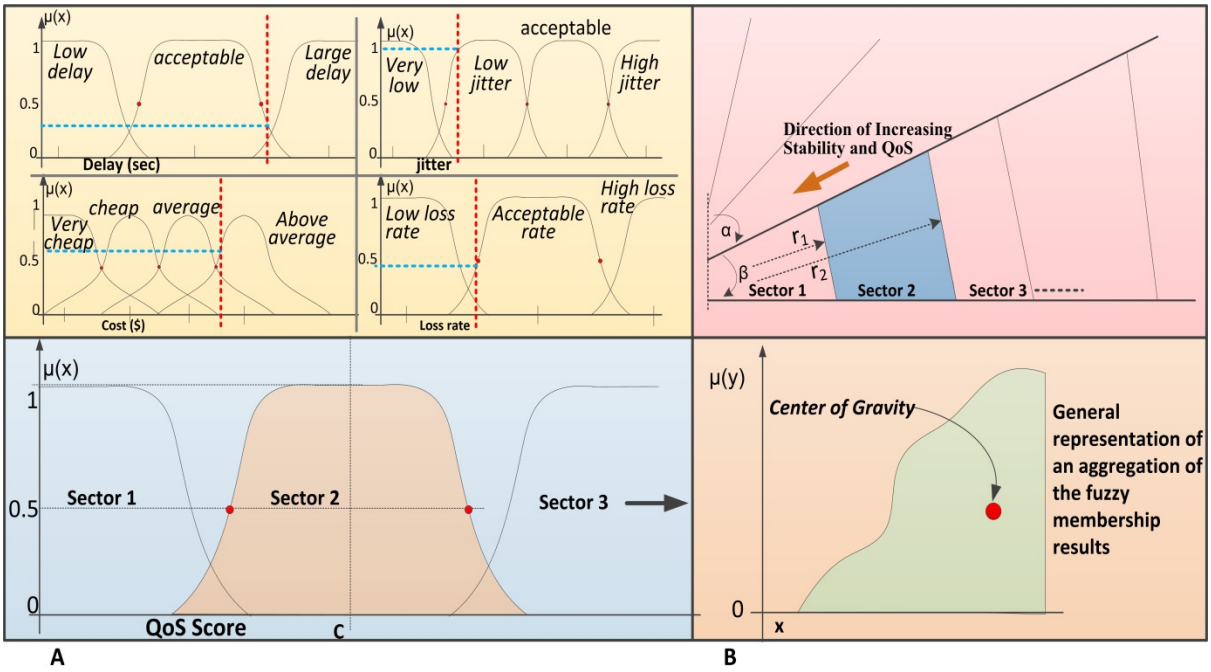


Figure 6.6 **A) Possible fuzzy membership functions for four QoS properties. Each presents varying membership parameter values and the MP's provided QoS level (red dotted line). B) Output membership function illustrating the hybrid SON sector to which the next query is directed.**

Thus using the fuzzy-generated score and the band-divided scores of the general hybrid SON, the searchable band(s) can be determined for the next-hop MP as visually illustrated in Figure 6.6. A variety of defuzzification methods can be used, however we based our system on the centroid method. Details pertaining to centroid defuzzification were discussed earlier in Section 6.1.

6.4. Fuzzy-Based Candidate MP Search

Once a valid MP sector band is obtained by the MS, reachable MPs must be forwarded a service composition request to begin forming an SSON. Message forwarding is directly related to the Hybrid SON introduced earlier. MPs receiving a composition request from a MS or another MP in the composition path must perform three actions:

a. Evaluate the semantic and syntactic similarity of its provided service compared to the plan-based description. Further details are provided in Chapter 7.

b. Evaluate and form a decision regarding the difference between the expected QoS level and that provided by the MP's service as discussed earlier.

c. Forward the request to nodes falling within the same MP sector band.

The way a MP forwards a message depends on its role within the Hybrid SON. MN nodes can only pass requests to their parent Major node. If the parent node falls within a higher-score band, forwarding is halted. NM nodes can forward requests in two directions: to neighboring nodes in the same Chord ring structure, and to attached MN (leaf) nodes if any. Accordingly, LM nodes can forward messages to the same set of nodes as NM nodes with the addition of the higher-level Chord ring to which it is linking.

We have opted for optimal Chordal rings [108] for each chord ring within our hybrid hierarchical SON. Our decision stems from the structures' symmetry, high fault tolerance, low broadcast time, and ease of routing. Each node in an optimal Chordal ring is connected to 4 nodes: two neighbor nodes, and two finger nodes.

Any MS or MP node receiving a composition request performs the algorithm in Figure 6.7 and Figure 6.8. The algorithm summarizes the steps of receiving a composition request, evaluating and comparing QoS levels through fuzzy logic, finding next-hop sector borders and band distance, and forwarding messages.

6.5. MP Ranking and Decision Making

Once a node accepts to join a composition, an acceptance message is delivered back to the sender. It also searches for the next-hop MP. Some of the contacted next-hop MPs may reply with an acceptance to the request forwarded to them. Consequently, the sender MP has to decide which candidate must be chosen for the composition. The decision process is based on the same fuzzy-enhanced logic and ontology-based similarity evaluation used in the previous step.

However, an additional ranking step is added. Nodes are ranked by the current MP on a scale of 0 to 1 where 1 is the highest ranking node. Ranks are evaluated according to how close the MP's service is to the service needed in the MS's plan. Additionally, ranking is based on the highest level of provided QoS and the lowest cost. Ranks for each replying service $S_i, where\ 0 \leq i \leq I$, are given by the following simple equation:

$$Rank_{S_i} = w_{near} Near^{Si} + w_{QoS} QoS^{Si} - w_{cost} Cost^{Si},\ such\ that\ 0 \leq Rank_{S_i} \leq 1 \qquad (6.14)$$

w_{near}, w_{QoS} and w_{cost} are the weights given to semantic nearness, QoS levels, and the service's total cost respectively. The total sum of these weights should be 1.

The MS chooses the highest ranking MP for composition. The chosen MP is informed of the decision and proceeds to repeat the process for the next-hop MP. The cycle continues until the MC is reached and a completed composition path has been established. A summary of the above process is illustrated in Figure 6.8.

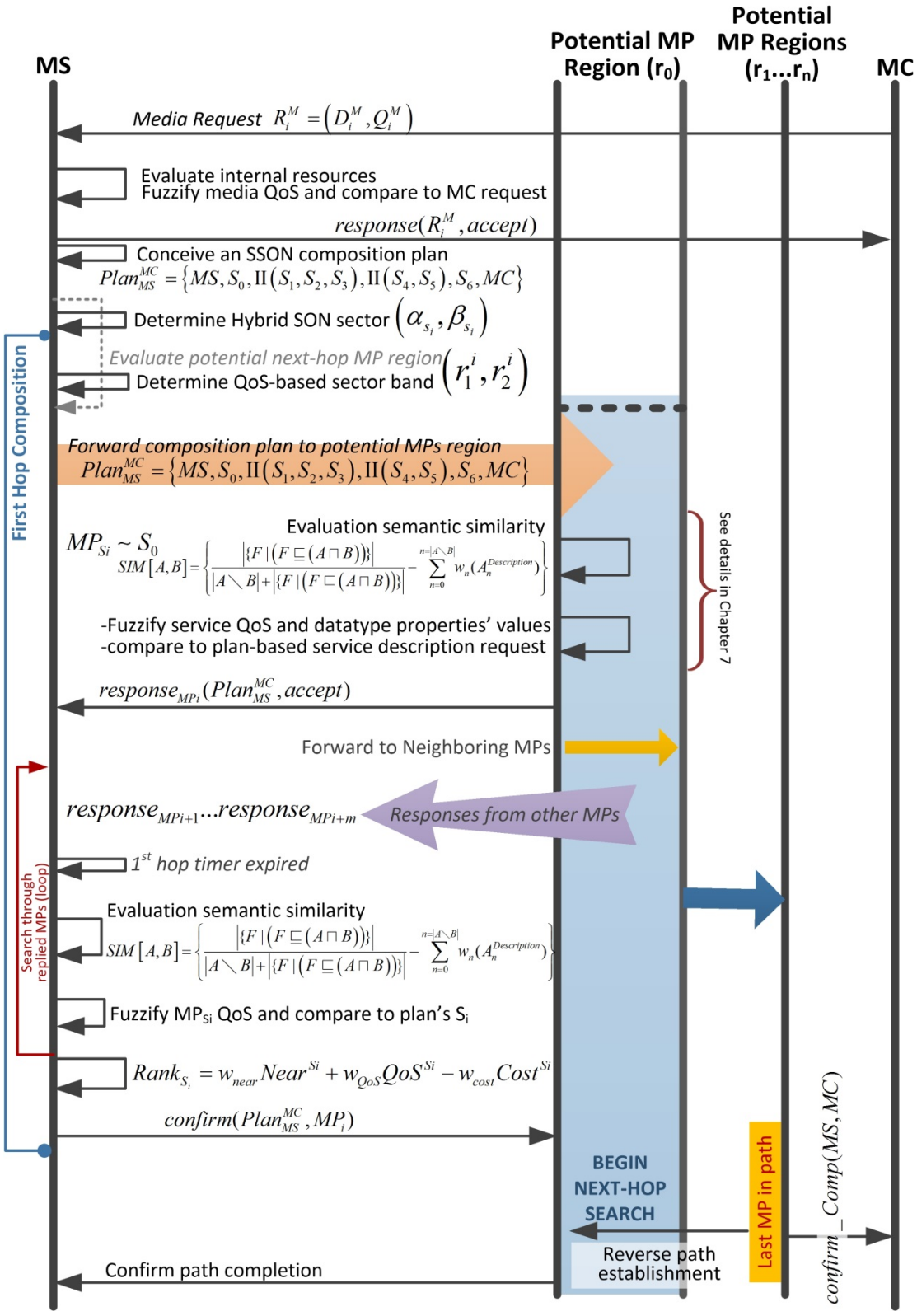


Figure 6.7 Timeline of exchange of messages between MC, MS, and MPs, and the functions performed regarding fuzzy QoS evaluation, and semantic nearness.

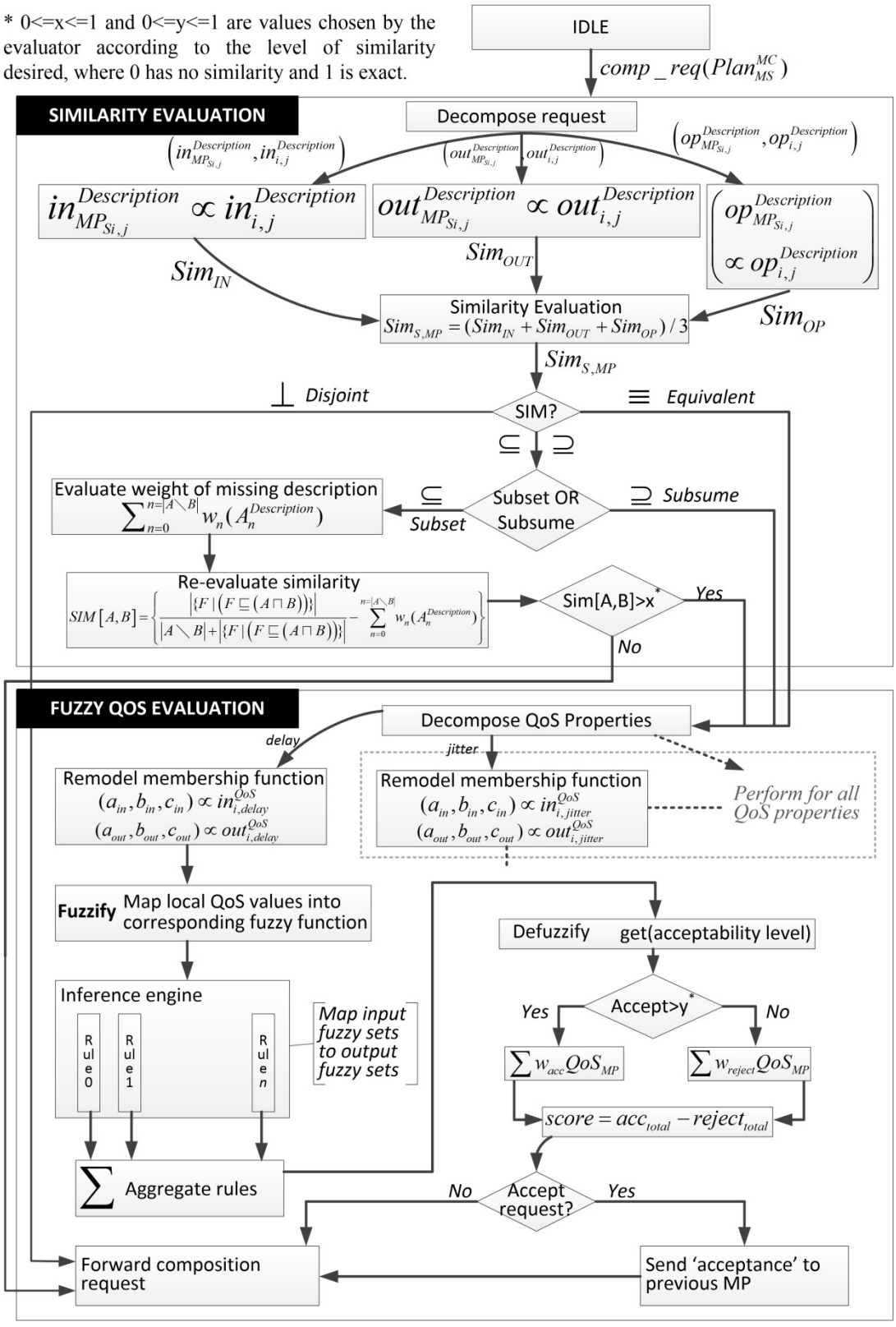


Figure 6.8 **State diagram of steps involved in evaluating internal QoS level compared to the plan description, and semantic nearness evaluation.**

6.6. Performance Evaluation

Our second stage of experimentation was focused on evaluating the performance of the fuzzy-based QoS-aware service composition engine. We utilized the open source jFuzzyLogic fuzzy engine to implement the fuzzy reasoning process [148]. It is a fuzzy logic package written in java and implements the fuzzy control language (FCL) specification [149]. Building a fuzzy inference system requires the generation of a control block obtaining input variables that control the following:

1. The MC-generated, or plan-based acceptable QoS levels used to evaluate the generalized bell- and sigmoidal-curve membership functions a, b, and c in (6.10) and (6.11).
2. The current functioning node's QoS properties' values.
3. Modifying the next-hop acceptable QoS given the possible degradation during the current step of the composition.

Additionally, the block diagram requires some output variables. In our case, the output variable represented a scoring value determining the acceptability of the QoS level based on the MC's or the plan's descriptions.

Fuzzy control in jFuzzyLogic is a rule-based expert system. The performance of the fuzzy inference engine is connected to the number of input variables, rules, and output variables used. In our simulation scenarios, only a single output per execution of the system was needed, the acceptability level of the MP's QoS. The number of rules was directly related to the number of input QoS properties being evaluated. As such, we have tested 4 different simulation scenarios with increasing numbers of QoS input properties and rules. The design was divided into 4 FCL template files that had from 1 to 4 QoS input properties. Each scenario involved 195 runs to smooth out the results for each test and remove any unwanted variations. Each run started with a random generation of hypothetical QoS properties' values, each of which was used to update the respective template file. The dynamic updates involved updating the fuzzy membership functions variables in (6.10) and (6.11), and the rules set conditions. The results of Figure 6.9 illustrate the total delay experienced in executing the fuzzy inference engine with increasing numbers of QoS

properties. Figure 6.9 shows that delay increases with each newly introduced property. In addition, it shows that varying the values of QoS properties, or the membership functions does not have an effect on the overall performance of the system; In other words, maintaining a specific number of QoS properties for an SSON composition process does not introduce delays with large standards of deviation. The results remain stable for tests involving the same number of input variables.

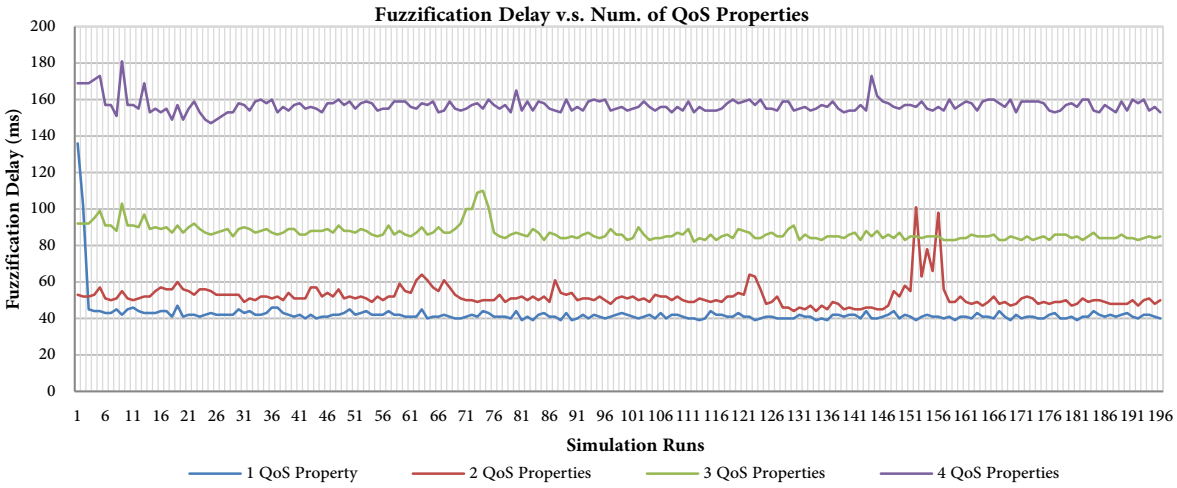


Figure 6.9 Delay experienced due to fuzzy inference engine with varying number of QoS properties inputs.

Consequently, the total delay experienced due to use of the fuzzy inference system can be generalized as follows:

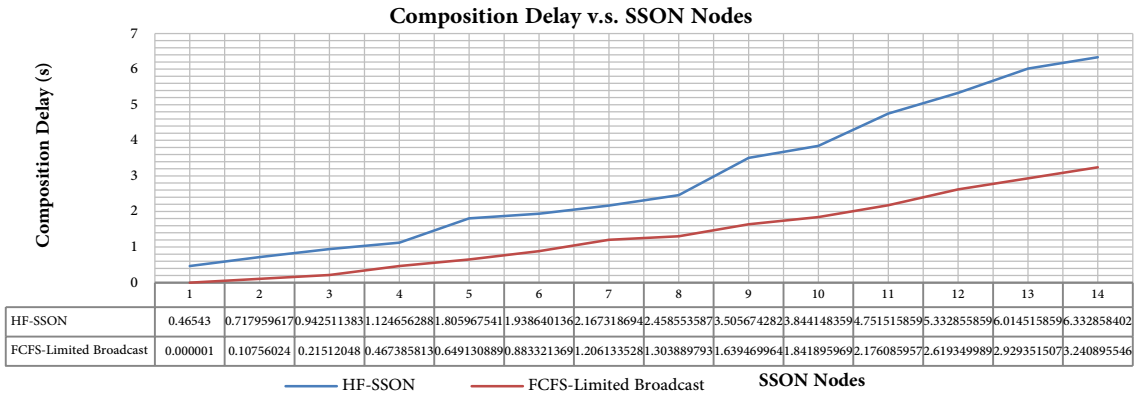
$$delay_{fuzzy} = delay_{internal}^{numParam} + numRep(delay_{external}^{numParam}) \quad (6.15)$$

where $delay_{internal}^{numParam}$ represents delay experienced within the MP itself due to evaluation of local QoS compared to the plan-based description. The delay is related to the number of input variables used: $numParam$. Additionally, next-hop candidate MPs replying to the composition request introduce additional delay depending on the total number of MPs that have replied, $numRep$, and the number of input parameters in each case. The observed delay ranged from 50 ms to 160 ms per MP.

Our next experiment involved combining our proposed hybrid SON, fuzzy inference engine, and semantic nearness evaluation methods to evaluate the performance of our proposed dynamic SSON composition method. The simulation involved 100 nodes randomly distributed and divided into 5% MCs, 10% MSs, and 85% MPs. The distribution of nodes attempted to model a network where the number of existing MPs can satisfy all incoming MC requests. Reductions in MP distribution will result in increases SSON composition failures. The purpose of this simulation is to test the delay experienced in successfully composing needed services. Each MP was associated randomly with a random set of services. Each service was provided at specific levels of QoS specified in a service description file. Similarly, each MS was associated with a random set of resources and a random set of media files. Arriving MCs were randomly associated with media specification files that defined the media to request. MCs were also associated with resource description files enumerating the required QoS levels for each media to be requested. Each MP was capable of supporting 1 to 4 parallel SSONs beyond which incoming composition requests were automatically rejected. Nodes were added to the network according to a Poisson process at a rate of 1 node/second. Similarly, network joins and failures were also modeled by a Poisson process with a mean arrival rate of 30 seconds.

Several types of messages were implemented with varying sizes. '*compositionRequestCall*' and '*compositionRequestResponse*' messages exchanged between the MC and MS were used to initialize the service composition process. The MC message included the identity of the required media and expected level of quality. The MS's response confirmed its acceptance or rejection of the request. The former message had a length of 64 bytes while the latter had a length of 32 bytes. The MS follows its response with a '*pathFormationCall'* to a set of potential MPs. MP search follows the sector-band search process discussed in Chapter 4.

Each MP forwards the message to potential MPs and those that accept the request reply with a 32-byte '*pathFormationResponse'* message to the MS. The MS sends a 64-byte '*pathFormationChosenCall'* message to the chosen MP. The message includes the full composition plan designed by the MS. The process of next-hop MPs searches continues until a 32-byt '*pathCompleted*" message is received by the MC from the last-hop MP.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14
HF-SSON	0.46543	0.717959617	0.942511383	1.124656288	1.805967541	1.938640136	2.167318694	2.458553587	3.505674282	3.844148359	4.751515859	5.332855859	6.014515859	6.332858402
FCFS-Limited Broadcast	0.000001	0.10756024	0.21512048	0.467385813	0.649130889	0.883321369	1.206133528	1.303889793	1.639469964	1.841895969	2.176085957	2.619349989	2.929351507	3.240895546

Figure 6.10 Composition delay with increasing composition path lengths for our H-SON and First Come First Serve approach.

Once the path is completed, a reverse confirmation '*pathEstablishmentCall*' message is sent informing each intermediate MP of the MPs ahead of it in the path towards the MC. This information becomes useful in the event a failure occurs in an intermediate MP and an SSON reconfiguration is required immediately. To maintain an updated view of the composed SSON, each node periodically runs stabilization routines at randomized intervals averaging 10 seconds. The stabilization routines maintain an updated link status for each MP's previous and next links in an SSON. If a failure is detected, the SSON path must be reconfigured.

Our simulation scenario involved performing comparison tests between our proposed Fuzzy-based SSON construction over the hybrid SON, referred to as HF-SSON henceforth, and the limited broadcast approach. Additionally, limited broadcast was extended into two types; The first is the normal limited broadcast where each node forwarded query messages to *n* neighbors. We have set *n=7* for our simulation purposes. The value for *n* has been chosen to mimic the same broadcast value utilized in the H-SON. Each MP searches for the next hop service by broadcasting its request to *n* nodes within the bounded search area of the hybrid overlay. In this scenario MPs blindly broadcast their queries to their neighbors in hope of finding a suitable next-hop MP. Forwarding scope was limited to 4 hops after which the query was dropped. Accepted query replies were taken first-come-first-served; the first replying MP was chosen for the next-hop composition. In the second scenario, limited broadcast was enhanced with our fuzzy-based MP ranking but not the semantic nearness

enhancement. As such, replying MPs were ranked according to their levels of provided QoS determining the best-fitting MP to be chosen for the next-hop composition.

Figure 6.10 shows the gap in composition speed differences between our proposed method and that of the normal limited broadcast. The delay increases steadily with the number of intermediate MPs required in the SSON path. The slowdown experienced in our method is due to FCFS-Limited Broadcast's *(First Come First Serve)* lack of evaluation for how precise the composition path will be in terms of function and QoS to the MC's requests. However, a large difference is expected between the two approaches in the amount of messages exchanged to form the composition path. Although limited broadcast attempts to avoid flooding the network, however, the number of messages exchanged during the composition phase far exceeds our approach of forwarding messages to specific sector bands in the overlay for each hop. Furthermore, by adjusting the search-scope of limited-broadcast, a flooding-like scenario can be achieved.

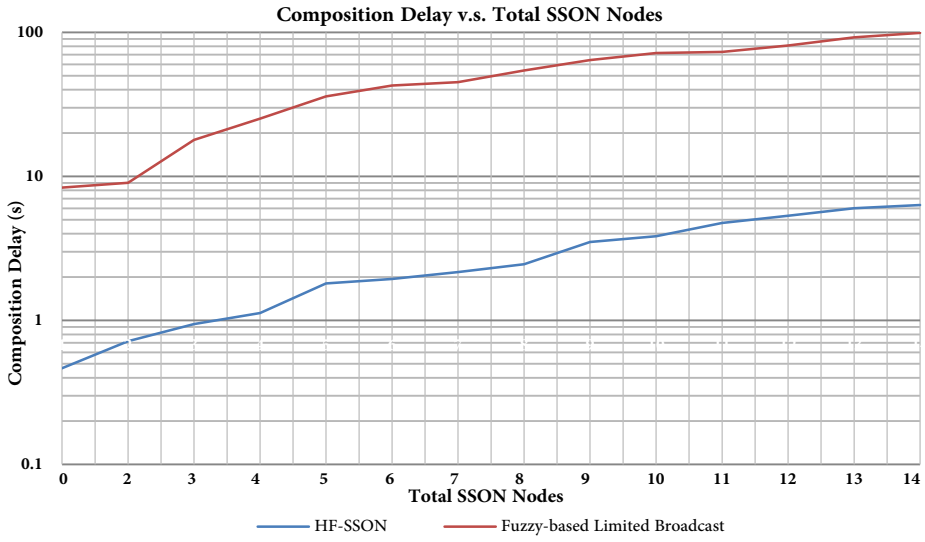


Figure 6.11 Composition delay experienced with increasing SSON composition path length for fuzzy-based limited broadcast and our composition method.

Nevertheless, enhancing FCFS with fuzzy-based QoS evaluation for better MP service selection accuracy, the composition delay performance was observed to be much higher than our presented approach. The benefits gained in increased QoS accuracy of chosen MP

services were overcome by an increase in delay. The test clearly displays the benefits of the H-SON in locating needed services with minimum delay even when fuzzy-based QoS and semantic similarity evaluations are involved in comparison to non H-SON methods. Composition tests for SSONs involving 1 to 14 intermediate MPs were performed for 30 simulation runs each and the average composition delay was recorded. As presented in Figure 6.11 an average of 10 fold increase in delay is observed in the fuzzy-based limited-broadcast when compared to our method. These results can be attributed to the fact a larger number of MPs are contacted per hop in the limited broadcast. Each node performs an internal fuzzy evaluation of its QoS level. Additionally, the previous MP must rank all received responding MPs. The larger number of MPs than our proposed method introduces a larger delay yet provides no increase in matching accuracy to the composition plan.

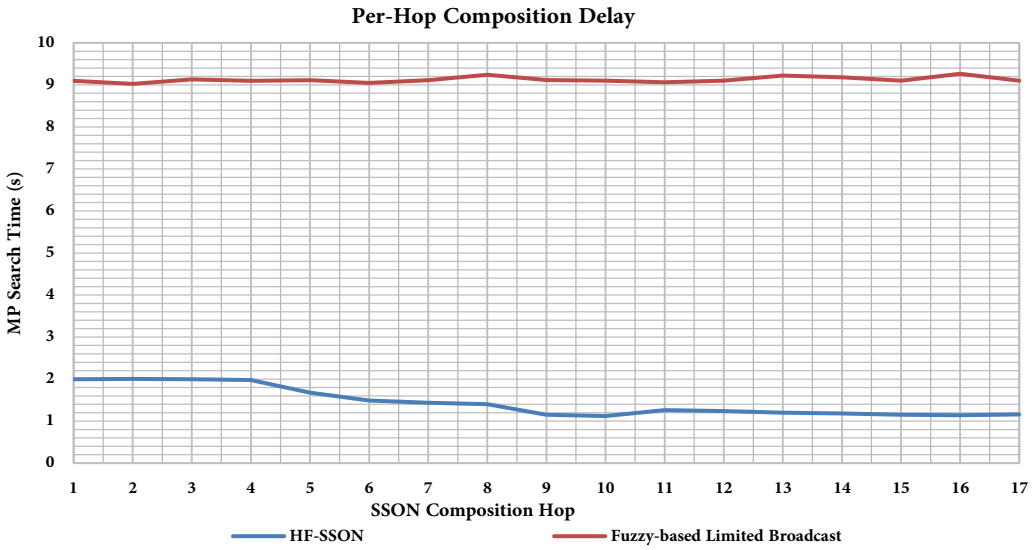


Figure 6.12- The per-hop composition delay experienced in a 17-MP long SSON.

Our final test involved measuring the delay experienced in forming each hop of the SSON composition shown in Figure 6.12. The fuzzy-enhanced limited broadcast maintained a steady level of delay for each composed hop. This is considering the fact that the number of MPs contacted is according to the values of *n*; the number of neighboring nodes, and the search scope. However, in our proposed method a significant decrease in per-hop composition delay is confirmed. This is in fact due to two reasons:

1. The use of sector-band search restrictions limits the number of MPs contacted in each hop.
2. Modifications performed to the variables $a, b, and\ c$ in (6.10) and (6.11) for each hop bring further reductions to the number of suitable MPs. The acceptable QoS levels set by the MC in its request become more restrictive as each intermediate MP beginning with the MS performs its respective component service. Since each service is provided at a level falling within the client's acceptable quality range, the next-hop MP must restrict its input QoS to a level that does not exceed that of its predecessor. Consequently, the number of potentially acceptable next-hop MPs converges to MPs that fall within close range of the MC's requirement. That is, nodes that meet or slightly exceed the high-priority QoS properties, while possibly slightly falling below acceptable QoS levels for low-priority properties.

Consistently our proposed QoS fuzzy-based semantic-similarity enhanced SSON construction over a hybrid SON provides performance levels that exceed the limited broadcast approach, guarantees a QoS level that meets the MC's requirements, and reduces the composition delay experienced.

6.7. Summary

This chapter presented a fuzzy logic based QoS evaluation method that can be utilized at three points in the SSON composition process. First, as an internal evaluation of the MP/MS's own level of quality compared to the requested services thus determining whether it should send an acceptance or rejection for the request. Second, fuzzy logic can be used to evaluate the distance away from the H-SON's center where potential next-hop MPs can be located. The process determines the borderlines for the highest and lowest acceptable QoS levels. Third, it can be used to rank potential next-hop MPs that have accepted to join the SSON path. From this set of MPs, the highest ranked MP is chosen as part of the path.

The next step in our research is to provide a clear approach for determining the validity of MP services chosen as part of an SSON. A semantic and syntactic nearness evaluation method is presented in Chapter 7. Through this approach, the input and output ports along

with the service's operations are semantically and syntactically compared to those required according the MS-generated composition plan. Those services with the highest nearness degrees represent the closest services to a match, and thus are chosen for composition.

Chapter 7

Plan-Free SSON Composition – *Utilizing Semantic Nearness*

7.1. Chapter Overview

Service composition is performed through the sequence of service discovery, combining and linking of discovered services, and executing the necessary components [150]. Consequently, the ability to efficiently and effectively select and integrate distributed services at runtime is an important step towards successfully composing user-oriented services. Service (or resource) discovery is comprised of two components: the discovery mechanism, and the MP candidate identification process. Various discovery mechanisms have widely been researched in the past [151] [152] [153]. Centralized approaches include a registry that maps all available resources with the nodes offering them. They represent the simplest solution, nevertheless they suffer from scalability, bottlenecks, and single point of failure problems. Some of the scalability issues are solved through Distributed Hash Table – based (DHT) approaches [154] [155] [156] [157] [158], however these approaches are limited to exact lookups. Moreover, large overheads are associated with DHT-based solutions especially in highly dynamic networks where persistent arrival and departure of nodes require continuous hashing and rehashing of distributed data. Direction-based discovery methods are another approach pointing service queries in the direction emanating from the MS towards the MC's [32]. Reductions in overhead are noticed; however discovery failures require expanding the search scope and restarting the discovery process.

The second component of service discovery is the mechanism utilized to identify successful candidates from contacted MPs. In the past, several initiatives have been conducted to provide languages and platforms that simplified the collaboration and integration of heterogeneous systems. Most developed standards focused on the integration of web services. These included the Universal Description, Discovery, and Integration (UDDI) standard [159], Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP) [98], and Business Process Execution Language for Web Service (BPEL4WS) [70]. Service composition problems take as input a set of component services, a composition goal, and generate a composite service represented by a workflow that achieves the desired goal. However, service composition mechanisms can face mismatches that appear at different levels of services such as their signature, behavior, quality of provided services and semantic descriptions.

Overcoming such mismatches has mainly relied on semantic and context matching-based solutions [160] [161] [162] [163]. However, most proposed solutions faced at least one of three problems. First, current industry standards for service description focus mainly on keyword-based matching [164] [165]. Interface descriptions of services are used to detect mismatches. Consequently, structural and behavioral mismatches can be utilized for both service compositions and adaptations [166]. The second problem is the reliance on a centralized entity to create an abstract composition model. This is also referred to as static composition. In static composition, the original abstract model has to be respected during the service composition and execution phases. However, static compositions do not have the flexibility associated with dynamic compositions that allow for atomic services to be automatically combined into a larger number of new services created from a limited set of service components. Dynamic composition has the ability to dynamically adapt to unpredictable changes in the environment. In the event of service failures or environmental changes in dynamic compositions, the services need to automatically reconfigure themselves without human or centralized intervention. However, dynamic service compositions are faced with several limitations: long response times, incorrect composition orders, interconnected service incompatibilities, and service unavailability.

We propose an autonomic semantic- and syntactic-nearness-based candidate MP selection algorithm for service compositions to automate the process of creating, optimizing,

and adapting SSONs. Service composition is fully distributed and is achieved without reliance on a central management entity. The algorithm is completely independent of the service discovery mechanism and focuses mainly on MP selection process. As such, our proposed work can be applied to most resource and service discovery mechanisms available by enhancing their MP selection process with greater accuracy and minimum additional latency.

The contributions of this chapter are: (1) a semantic similarity service comparison method used to evaluate whether a service a) matches the semantic description of the task required according to the composition plan, or b) has an input port that matches the output port of the previous hop service in our plan-free SSON composition model; (2) a fully distributed and dynamic service selection mechanism that composes autonomic MPs into SSONs; (3) a candidate MP selection algorithm that is independent of the discovery mechanism giving the system greater flexibility and adaptability; (4) QoS-based MP selection restricted to services that meet the MC-initiated requirements; and (5) utilization of semantic and syntactic nearness measurements to evaluate the benefits of choosing certain candidate MPs thus avoiding incorrect composition orders and service incompatibilities.

The remainder of this chapter is organized as follows. Section 7.2 presents a semantic similarity evaluation method. Section 7.3 presents some of the related work pertaining to plan-free service composition and semantic-based service discovery and composition. Section 7.4 summarizes some of the system requirements. Section 7.5 presents the ontology-based semantic nearness evaluation utilized in composing SSON without reliance on a MS-generated plan. Section 7.6 discusses our simulation scenarios and evaluations of our system.

7.2. Semantic Similarity

The decision made by each MP receiving a composition request, in the plan-based composition process discussed up to this point, involves evaluating the semantic and syntactic similarity of its provided services in comparison to service descriptions given in the MS's composition plan $Plan_{MS}^{MC}$. The similarity value is needed to rank and choose the most-

fit service from candidate services in (6.14). Extensive research has been conducted in regards to semantic comparison between web services [167] [168] [169]. Autonomic composition of services involves the need to support some degree of semantic evaluation of such services. Ontologies play an important role in the semantic-driven service composition. It is reasonable to use semantic distance to calculate the distance between concepts because ontologies constitute trees [170] [171] [172]. In this case distance refers to the number of tree links between two ontology classes or object properties. The distance and branching of these links determines whether concepts are related or disjoint.

Every MP receiving a composition request must evaluate the semantic nearness of its provided service compared to that described in the plan. Similarly, a MP receiving acceptance replies from candidate next-hop MPs should be able to determine the most-fit MP for the composition to proceed. Semantic similarity evaluation is applied at three sections of a component service; its input port, its performed service, and its output port as follows:

$$Sim_{IN} = SIM\begin{bmatrix} In_{MP_{Si}}\left(in_{MP_{Si,j}}^{Description}, in_{MP_{Si,j}}^{QoS}\right), \\ In_{i,j}\left(in_{i,j}^{Description}, in_{i,j}^{QoS}\right) \end{bmatrix} \tag{7.1}$$

$$Sim_{OUT} = SIM\begin{bmatrix} Out_{MP_{Si}}\left(out_{MP_{Si,j}}^{Description}, out_{MP_{Si,j}}^{QoS}\right), \\ Out_{i,j}\left(out_{i,j}^{Description}, out_{i,j}^{QoS}\right) \end{bmatrix} \tag{7.2}$$

$$Sim_{OP} = SIM\begin{bmatrix} Op_{MP_{Si}}\left(op_{MP_{Si,j}}^{Description}, op_{MP_{Si,j}}^{QoS}\right), \\ Op_{i,j}\left(op_{i,j}^{Description}, op_{i,j}^{QoS}\right) \end{bmatrix} \tag{7.3}$$

The similarity values $Sim_x, x = \{IN, OUT, OP\}$ range from 1 for an exact match, to 0 for disjoint concepts. Similarity in its general form is divided into five categories [173]:

1. **Exact** (≡)**:** If $T \vDash n_{MP_{Si,j}}^{Description} \equiv n_{i,j}^{Description}$ $where\ n = \{in, out, and\ op\}$. Meaning every concept found in the plan-based description is found in the candidate MP service.
2. **Subset** (⊑)**:** if service description is a sub concept of the desired service, $\forall n_{i,j}^{Description} \Rightarrow \forall n_{MP_{Si,j}}^{Description}$. This occurs when the candidate MP's description (classes, properties) occur at a lower level in the ontological tree than that of the plan-based description. This is valid only if both descriptions occur within the same

ontology branch. Otherwise the two descriptions are disjoint. A MP description that is a subset of the desired description is usually rejected. However, depending on the property's priority level the MP can be accepted.

3. **Subsume** (⊒)**:** when all MP service description embodies all of the plan-based description and is a super concept of it. That is, $\forall n_{MP_{Si},j}^{Description} \Rightarrow \forall n_{i,j}^{Description}$. Consequently the candidate MP covers all the required descriptions as stated within the plan.

4. **Intersect** (∩)**:** if some of the MP's service concepts are equivalent to some of the service concepts declared in the plan. However, some of the concepts are missing from the candidate MP service. Therefore $\exists n_{MP_{Si},j}^{Description} \Rightarrow \exists n_{i,j}^{Description}$.

5. **Disjoint** (⊍)**:** if none of the MP's service concepts are equivalent to any of the service concepts declared in the plan. Therefore $\overline{\exists n_{MP_{Si},J}^{Description}} \Rightarrow \exists n_{i,j}^{Description}$. Such MP service is automatically rejected.

The similarity function of (7.1) when comparing the input port descriptions of the plan-based service Si and the candidate MP service MP_{Si} is given by the following relationship:

$$let\ A = \ \forall\, x \in in_{i,j}^{Description} | 0 \le j \le n \tag{7.4}$$

$$let\ B = \ \forall\, y \in in_{MP_{Si},j}^{Description} | 0 \le j \le m \tag{7.5}$$

$$SIM[A,B] = \left\{ \frac{|\{F|(F \sqsubseteq (A \sqcap B))\}|}{|A \backslash B| + |\{F|(F \sqsubseteq (A \sqcap B))\}|} \right\} = \begin{cases} 0 & if\ A \perp B \\ 1 & if\ A \equiv B \\ < 1 & if\ A \sqsubseteq B\ or\ A \sqsupseteq B \end{cases} \tag{7.6}$$

7.3. State of the Art Research

Semantic approaches have been proposed in the past as a means for enhancing web service search mechanisms [174] [175] [176] [177]. This was later utilized to cluster peers with similar content to become part of the same Semantic Overlay Network (SON). Such clustering simplified communication costs and increased query accuracy levels. However,

successful composition processes are highly dependent on the mechanism for filtering and selection of services. As such, the dynamic composition of services requires an understanding of the capabilities and compatibilities of services. Full automation of composition is still an object of ongoing research, while partial automation of composition has had more success in the past. With the semantic web technology full automation becomes more feasible. Through semantics, information is given well-defined meaning thus better enabling computers and people to collaborate. Sirin et al. [169] demonstrated a goal-oriented, interactive composition approach with a matchmaking algorithm to help users filter and select services while building compositions. The proposed system utilized OWL-S service descriptions (*ServiceProfiles*) to aid in the composition process. When a service needs to be located, the system creates a ServiceProfile for the desired final service. A service registry matches the requested profiles to those advertised through subsumption. The matching value is then classified according to four classes: exact, plugin, subsume, or fail. A composer component is then responsible for choosing the best services for composition.

The above work suffers from three limitations: First the reliance on a centralized service registry component. This presents a problem when applied to highly dynamic mobile environments where no central control point may exist [178]. Second is the need to generate a composition workflow (plan) that lists the requested profiles. Generation of the workflow abstract relies on a centralized entity and limits the systems independence in choosing component services for a composition. Third is the oversimplified matchmaking process. Limiting the matching classification to only four classes as well as omitting the importance of QoS during service selection negatively affects the usefulness of the proposed composition method. This in fact is similar to the plan-based semantic similarity enhanced composition we have presented in Chapter 6 along with Section 7.2 of this chapter.

The issue of service discovery given non-explicit service description semantics is confronted by Paliwal et al. [173]. The work involves semantic-based service categorizations and semantic enhancements of service requests through an ontology framework. It utilizes service selection based on semantic service descriptions rather than syntactic keyword matching. However, service classification is performed offline through UDDI, furthermore relying on a centralized registry. Categorization of services is followed by service selection from relevant clusters. This is achieved by parameter-based service

refinement that narrows the set of appropriate services matching the service request based on service parameters (input, output, description). Service descriptions are used along with ontological concepts to generate semantic descriptive vectors of the services. After, clustering of service vectors is performed where each cluster is associated with an ontology. Services are then chosen according to the similarity level of their descriptions in comparison to the service request. The evaluation employs a widely used LSI-based technique that uses cosine measures as a similarity metric. The work presented is highly centralized, requires offline registration of service descriptions, and does not clearly specify how QoS is taken into account. Consequently, the solution is impractical for mobile networks.

Wang proposed to achieve dependable service composition by taking mobility prediction of service providers into consideration [179]. His solution utilizes service providers' abilities to predict their duration of stay within the network. To handle the uncertainty associated with location prediction, the authors use models to characterize the uncertainty: a probability-free model and a probabilistic model. The former refers to service composition when the uncertainty of the mobility prediction is given by a time window. The latter is used when the estimated time that each service provider can stay in the current environment can be accurately modeled with known probability distributions. However, the system's mobility prediction information is disseminated through UDDI protocol. Consequently when a service node moves to a new environment it must register its services with the closest service repository. Moreover, the service node must also indicate the estimated time during which it will be present at the current environment. This centralized approach to service discovery is inefficient for dynamic mobile networks as discussed earlier.

An agent-based automatic service composition solution was proposed by Tong et al. [12]. The solution integrates service and software agent technologies into one cohesive entity. Three models are utilized: belief, action, and plan models. The set of beliefs denoting the knowledge of a service agent about itself and the environment is encapsulated by the beliefs model. Beliefs include general facts known by the agent, relationship constraints with other service agents, and social knowledge concerning information known about the capabilities and addresses. The action model focuses on internal service actions, communication, and operation templates characterizing the actual operations with similar functions provided by other services. Finally, the plan model encapsulates the business logics of how to use

operation templates to achieve certain business goals. Plans specify the logic relationships among the operation templates and the data flows indicate the types of message dependencies among operation templates in the plan. The reliance on plans in the system requires an initial offline human intervention to generate the required plans. Furthermore, the static nature of plans used implies the system's inability to adapt to highly dynamic environments. Moreover, the lack of consideration for QoS diminishes the system's abilities to meet the MC's requirements.

Fujii et al. [75] presented an architecture that permitted dynamic and semantic-based composition of complex services. The architecture obtained the semantics of the requested services in an intuitive form and dynamically composed the services bases on their semantics. A workflow of the requested service was created using discovered components, and the workflow was then executed. The system relied on a middleware named Component Runtime Environment (CoRE) that provided the service discovery and execution functionality. Thus, this system faced the same problems other centralized plan-based composition solutions had faced [135] [136] [180] [181].

7.4. Problem Summary of Plan-Free Composition

7.4.1. System Requirements

To empower SSON composition mechanisms with full autonomy, a shift is required from plan-based to plan-free compositions. Our former approach presented in Chapters 5-6 relies on a MS-generated plan. The plan clearly models all intermediate MP services required in the path from the MS to the MC. Intermediate MP queries depend upon a semantic evaluation pertaining to "syntactic similarity" between the plan-based task description and the descriptions of discovered MPs. Semantic similarity under such circumstances refers to how identical the service description is in relation to the abstract task descriptions included in the plan. It is an ontology-based property match-making process that determines ontology properties shared between the plan-based tasks and those missing from queried services. According to the weights i.e. importance of missing properties a decision is made to accept or reject joining the SSON path. Similarity is evaluated at three levels of a service MP:

input port, output port, and service operation. This results in several similarity criteria (values) that are utilized to rank candidate services. A final decision is made such that the highest ranked MP is chosen. Details were discussed in Section 7.2.

However, the above process is loosely centralized. The MS is responsible for evaluating the number and types of intermediate MPs required to complete the SSON composition and form a plan template for MP discovery. Although the query and decision-making processes for each consecutive MP in the SSON path is autonomously performed by the previous-step MP, however the decision-making process is performed according to the original MS-generated plan-based guidelines.

In this section, we introduce an SSON composition process that enables every intermediate MP to have the flexibility in determining the type and identity of the next-hop MP without relying on a composition plan. As such, every SSON MP member must formulate a one-step composition decision from a set of candidate MPs through the utilization of *semantic and syntactic nearness*. Our goal is to be able to reach a successful SSON composition while respecting the following conditions:

1. A clear description of MC requirements in terms of media flow type and QoS is provided by the MC to initiate SSON composition.
2. A plan-free solution. The current MP that is member of the composition path is unaware of the type and quality level of the required next-hop service required. Hence the system's inability to rely on a service cluster-based underlays as presented in [182] [183] [184] [185] [186], or the H-SON presented in Chapter 4.
3. A lack of abstract task descriptions. It is impossible to compare descriptions of discovered MP services to any abstract models.
4. Prevent the creation of unnecessary loops in the SSON path. The composition should advance logically and functionally forward with each hop. As such, the SSON will essentially avoid any unneeded loops.

7.4.2. Plan-Free SSON Composition Steps

Composition solutions providing autonomic, QoS-based services require an automatic service discovery method. For services to be automatically discovered and invoked without human intervention, and for MCs, MPs and MSs in the network to operate together, a formal description of requests and services is required. Human interaction greatly limits the scalability of solutions and the composition's performance. MC service request models and service description models were discussed earlier in Sections 5.3 and 5.4.

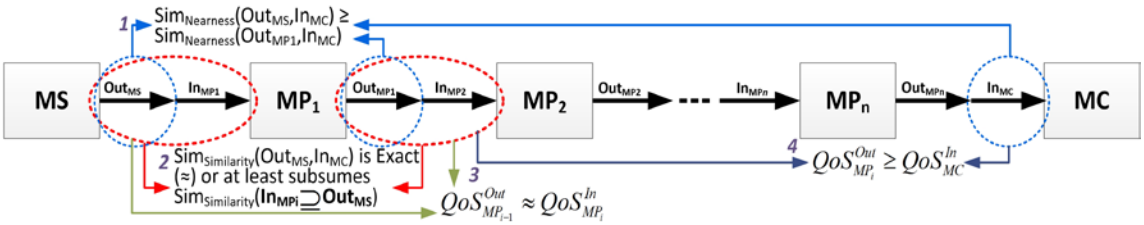


Figure 7.1 An abstract SSON path illustrating the four processes to be performed at each hop for a successful plan-free service path composition.

We need a MP service discovery method that can bring, over the course of successive linking, the SSON composition process as close as possible to the MC requirements. It is known that at each MP in the SSON composition path two types of information are available to us:

- *MC Requirements*: represents the initial MC media stream request. As illustrated in Section 5.3 the MC sends a media request that includes the type and identity of the requested media stream and the acceptable levels of QoS properties. As such, the borderline between acceptable and unacceptable QoS values is well established.
- *Received Stream Description*: this is the stream received by the current MP from the previous-hop MP. Any degradation in QoS from the initial MS stream is thus visible. Information such as percentage of loss, jitter, delay, etc. can be deduced from the media stream. The current MP can alternatively be directly informed of such information from the previous-hop MP.

The main idea behind our proposed plan-free SSON composition process derives its conceptuality from the term *arrow of time* [187]. The term was coined by the British astronomer Arthur Eddington to describe the one-way direction of time. Consequently, if time was perfectly symmetrical then real events would seem realistic whether viewed forwards or backwards. A concept of particular interest to our work is that of measuring the semantic disorder or distance away from the MC's requirements as the system advances through time. Our goal is to formulate an SSON composition process that reduces the semantic distance (or increases the semantic nearness) between the current hop of the SSON path and the MC's requirements. As such, service composition begins with a state of disorder and moves towards an ordered state. The former refers to the point when all component services are randomly distributed in the physical space (network), while the latter refers to the point when all required component services have been linked (ordered) to meet the MC's requirements. The change in semantic nearness $\Delta Sim_{Nearness}$ can be calculated to relate the current semantic state with that required by the MC compared to that of the previous SSON hop.

$$\Delta Sim_{Nearness} = Sim^{t+1}_{Nearness} - Sim^{t}_{Nearness} \tag{7.7}$$

A positive change represents a beneficial move towards order/meeting the MC requirements. While a negative change represents a disadvantageous move away from order or meeting the MC requirements.

Figure 7.1 illustrates the steps required to successfully construct an SSON that meets the MC's media flow needs and QoS requirements. We begin by declaring essential variables required to model our proposed solution:

Let $\boldsymbol{S^{Descr}_{current\_Out}}$ be the current service output description such that:
$\boldsymbol{S^{Descr}_{current\_out}} = \{\boldsymbol{s^{Descr}_{property_0}}, \boldsymbol{s^{Descr}_{property_1}}, \dots, \boldsymbol{s^{Descr}_{property_{m-1}}}\}$, where m represents the number of descriptive properties.

Let $\boldsymbol{S^{QoS}_{Current\_Out}}$ be the QoS level of the current MP service such that:
$\boldsymbol{S^{QoS}_{current\_out}} = \{\boldsymbol{s^{QoS}_{property_0}}, \boldsymbol{s^{QoS}_{property_1}}, \dots, \boldsymbol{s^{QoS}_{property_{n-1}}}\}$, where n represents the number of QoS properties.

Let $\boldsymbol{S^{Descr}_{next\_In}}$ be the acceptable next MP service input description such that:
$\boldsymbol{S^{Descr}_{next\_In}} = \{\boldsymbol{s^{Descr}_{property_0}}, \boldsymbol{s^{Descr}_{property_1}}, \dots, \boldsymbol{s^{Descr}_{property_{m-1}}}\}$, where m represents the number of descriptive properties.

[CONTINUED ON P.156]

[CONTINUED FROM P.155]

Let $\boldsymbol{S_{next\_In}^{QoS}}$ be the QoS level of acceptable media streams arriving at the the next MP: $\boldsymbol{S_{next\_In}^{QoS}} = \{\boldsymbol{s_{property_0}^{QoS}}, \boldsymbol{s_{property_1}^{QoS}}, \dots, \boldsymbol{s_{property_{n-1}}^{QoS}}\}$, where n represents the number of QoS properties.

Let $\boldsymbol{S_{next\_Out}^{Descr}}$ be the description of the output of the next MP service such that: $\boldsymbol{S_{next\_Out}^{Descr}} = \{\boldsymbol{s_{property_0}^{Descr}}, \boldsymbol{s_{property_1}^{Descr}}, \dots, \boldsymbol{s_{property_{m-1}}^{Descr}}\}$, where m represents the number of descriptive properties.

Let $\boldsymbol{S_{next\_Out}^{QoS}}$ be the QoS level of the candidate next−hop MP service such that: $\boldsymbol{S_{next\_Out}^{QoS}} = \{\boldsymbol{s_{property_0}^{QoS}}, \boldsymbol{s_{property_1}^{QoS}}, \dots, \boldsymbol{s_{property_{n-1}}^{QoS}}\}$, where n represents the number of QoS properties.

Let $\boldsymbol{S_{MC}^{QoS}}$ be the QoS level required by the MC in its original media stream request sent to the MS such that: $\boldsymbol{S_{MC}^{QoS}} = \{\boldsymbol{s_{property_0}^{QoS}}, \boldsymbol{s_{property_1}^{QoS}}, \dots, \boldsymbol{s_{property_{n-1}}^{QoS}}\}$, where n represents the number of QoS properties.

Let $\boldsymbol{S_{MC}^{Descr}}$ be the description of the media stream requested by the MC such that: $\boldsymbol{S_{MC}^{Descr}} = \{\boldsymbol{s_{property_0}^{Descr}}, \boldsymbol{s_{property_1}^{Descr}}, \dots, \boldsymbol{s_{property_{m-1}}^{Descr}}\}$, where m represents the number of descriptive properties.

Given our main goal of providing a plan-free autonomic SSON composition process, Figure 7.1 illustrates four evaluation components required to successfully compose a service:

1. A *semantic similarity* measure between the output of the currently chosen MP ($S_{current\_Out}^{Descr}$) and the input of the next-hop candidate MP ($S_{next\_In}^{Descr}$).
2. A *semantic nearness* measure that compares the similarity level between the output of the currently chosen MP ($S_{current\_Out}^{Descr}$) and the MC's requirements (S_{MC}^{Descr}), versus the output of the next-hop ($S_{next\_Out}^{Descr}$) candidate MP and the MC's requirements.
3. *Current QoS measures* between the output stream of the currently chosen MP ($S_{Current\_Out}^{QoS}$) and the input of the immediate next-hop candidate MP ($S_{next\_In}^{QoS}$), and
4. *Expected QoS measures* between the output stream of the next-hop candidate MP ($S_{next\_Out}^{QoS}$) and the MC's QoS requirements (S_{MC}^{QoS}).

Evaluation components 3 and 4 in Figure 7.1 illustrate the need to evaluate the variations in QoS levels between various MP services. However, exact QoS values are difficult to

obtain. The accuracy levels of data transmissions of media streams exchanged between MPs or jitter levels are subjective concepts. Values of nonfunctional properties are dynamically variable and are subject to the MPs' available resources. Values are also subject to the current loads on MPs depending on the number of SSONs they have joined and the types of services they provide. Consequently we must employ an approximate QoS evaluation process rather than exact. For this, fuzzy logic is utilized. The details pertaining to fuzzified QoS comparison was presented in Section 5.3. Semantic similarity measures comparing the output of the currently chosen MP and the input of the next-hop candidate MP were also presented earlier in Section 7.2.

7.5. Ontology-Based Semantic Nearness

7.5.1. Evaluating Semantic Nearness

The goal behind semantic *nearness* is to bring the SSON composition process closer to the MC's requirements with each hop. Consequently, if the semantic similarity between the current MP and the final result compared to the previous MP and the final result is greater, then it can be said that the SSON composition is successfully progressing in the correct semantic direction. Nearness is a property characterized by human perception and intuition, however a numerical quantification of the concept is plausible. Measuring semantic nearness between concepts is an important component in extracting any existing relationships, disambiguation, and determining service compatibility. Semantic similarity or nearness measures can be of great benefit in integrating multiple services for SSON composition. Semantic-based measurements depend on semantic descriptions of services and formulating the necessary semantic similarity evaluation rules.

Ontologies are critical in evaluating the semantic nearness level between service ports. Evaluating semantic nearness between ontology concepts has been widely researched and a number of evaluation methods have been proposed [188] [189] [190] [191] [192] [193]. Similarity measuring techniques can be classified into two classes: 1) structure (or graph)–based measures that use ontology hierarchy structures to compute the semantic similarity

between terms, and 2) information content IC (or term frequency) based measures that use IC statistics. In general, semantic *nearness* is affected by four factors: semantic distance, depth, coincidence, and density [194].

The most widely used method to semantic similarity between two terms in an ontology is to find the shortest path length between them. *Semantic depth* is defined as the number of concept nodes that are included in the longest path from the concept node to the ontology's root conceptual node. Wu and Palmer [195] used a variation of shortest path length that also utilized the depth of the lowest common ancestor (LCA) of any two concepts. The shortest path distance is scaled by the larger of the two concept paths. The similarity score thus depends on the relative positions of the concept pairs. We extend [195] to evaluate the *semantic distance* for two media stream descriptions having *J* conceptual properties: the output stream of the current MP and the required MC media stream. Then from (7.3) and (7.6), the length-based similarity is as follows:

$$sim_{Length}\left(OUT_{MP_i}, R_{MC}\right) = \left[\sum_{j=0}^{j=J} \max\left(\frac{2\times depth\left[LCA\left(out_j^{Description}, D_j\right)\right]}{len\left(out_j^{Description}, D_j\right)+2\times depth\left[LCA\left(out_j^{Description}, D_j\right)\right]}\right)\right]/J \quad (7.8)$$

where $depth(x)$ is the depth of the LCA of the two conceptual properties. The shortest path distance between the two conceptual properties is represented by $len(x)$.

Semantic coincidence is the ratio of the number of intersecting ancestor nodes to those in union of the two concepts being compared including the concepts themselves. If $p\left(out_j^{Description}\right)$ denotes the collection of ancestral concept nodes of conceptual property $out_j^{Description}$, and $p\left(D_j\right)$ is the collection of D_j property's ancestors, then semantic coincidence is given by the following:

$$sim_{Coin}\left(OUT_{MP_j}, R_{MC}\right) = \left[\sum_{j=0}^{j=J}\left(\frac{\left|p\left(out_j^{Description}\right)\cap D_j\right|}{\left|p\left(out_j^{Description}\right)\cup D_j\right|}\right)\right]/J \quad (7.9)$$

Finally, *semantic density* represents the local sparseness degree of concept partitions in an ontology hierarchical structure. It is defined as the ratio of the number concept nodes that exist from the layers of the two concept nodes being compared to the layer of the LCA node not including the LCA node itself, $N(x)$, to the number of layers which the nodes cross denoted by $d(x)$.

$$sim_{Density}\left(OUT_{MP_j}, R_{MC}\right) = \left[\sum_{j=0}^{J}\left(\frac{N\left(out_j^{Description}, D_j\right)}{d\left(out_j^{Description}, D_j\right)}\right)\right]/J \quad (7.10)$$

A number of similarity measurement methods have also been proposed and can also be utilized to evaluate semantic nearness levels between ontology concepts. A comprehensive semantic similarity (nearness) calculation method is obtained by combining equations (7.8), (7.9), and (7.10):

$$SIM_{Nearness} = f\left(sim_{Length}, sim_{Coin}, sim_{Density}\right) = w_A sim_{Length} + w_B sim_{Coin} + w_C sim_{Density} \quad (7.11)$$

such that $w_A + w_B + w_C = 1$. We simplify the process by taking the average of the three evaluation methods to have semantic nearness be equal to:

$$SIM_{Nearness} = \left(sim_{Length} + sim_{Coin} + sim_{Density}\right)/3 \quad (7.12)$$

As such a composition query received by MP_i is accepted if the semantic nearness of its output stream exceeds the nearness level of MP_{i-1} when compared to the MC requirements. This is summarized in (7.13).

if $SIM_{Nearness}\left(OUT_{MP_i}, R_{MC}\right) > SIM_{Nearness}\left(OUT_{MP_{i-1}}, R_{MC}\right) \rightarrow$ **ACCEPT**, Otherwise $\rightarrow$ **REJECT** (7.13)

An improvement to the semantic nearness evaluation of two service ports is related to the question of how the relative importance of relationships found between description properties can be determined. Anyanwu et al. [196] introduced three parameters that can help improve the accuracy of the above semantic nearness evaluation equations. These parameters include relevance, specificity, and the span of the relationship existing between two ontology concepts.

Two concepts can be associated with each other with reference to multiple domains. Each of these domains can be specific to certain services or user applications. Furthermore, each domain of a particular concept is itself a higher-level concept in the same ontology. For example, the ontology classes of *LossRate* and *Accuracy* can be associated with the *NetworkTransmission* and *MediaStream* upper ontology classes simultaneously. Consequently relevance can be used to represent the contextual relationship between any two ontology concepts. Thus, given two concepts c_i and c_j, then $Rel\left(c_i, c_j\right)$ evaluates the

contextual relationship between the two concepts. As such, $Rel(c_i, c_j)$ is equal to 1 if the two concepts are linked, and is equal to 0 otherwise.

The position of a concept in the ontology hierarchy can also have an effect on the ranking process. The position of a concept determines its specificity level. Concepts that are in the lower level of the hierarchy are considered more specific than higher level concepts which are seen as more generic. Thus given any two concepts c_i and c_j, then $Spec(c_i, c_j)$ evaluates the specificity level. The predicate evaluates to 1 if there is a downward path from c_i to c_j in the ontology. In other words, if c_i is an ancestor of c_j then specificity is equal to 1. In such case c_i becomes a more specific concept of c_j. Otherwise specificity evaluates to 0.

Concepts in ontologies can have, in addition to vertical divergence, horizontal divergence. This is referred to as the span of the relationship, and it expresses the strength of linkage among concepts. Span is used to restrict the scope existing between any two concepts. Furthermore, span only includes concepts at the same peer level of the two ontology concepts being compared. If the concepts are linked within the specified span, then the value of the span, $Span(c_i, c_j)$, is equal to 1, else it is set to 0.

Merging all three concept ranking methods yields the following equation:

$$Rank(c_i, c_j) = w_{rel} Rel(c_i, c_j) + w_{spec} Spec(c_i, c_j) + w_{span} Span(c_i, c_j) \qquad (7.14)$$

The above three ontology concept ranking methods for relevance, specificity, and span can be used to improve the accuracy of semantic nearness evaluation as presented in (7.8), (7.9), and (7.10) as follows respectively:

$$sim_{Length}(OUT_{MP_i}, R_{MC}) = \left[\sum_{j=0}^{j=J} \max\left(\frac{2\times depth\left[LCA\left(out_j^{Description}, D_j\right)\right]}{len\left(out_j^{Description}, D_j\right)+2\times depth\left[LCA\left(out_j^{Description}, D_j\right)\right]}\right) \times Rank\left(out_j^{Description}, D_j\right)\right]/J \qquad (7.15)$$

$$sim_{Coin}\left(OUT_{MP_j}, R_{MC}\right) = \left[\sum_{j=0}^{j=J}\left(\frac{\left|p\left(out_j^{Description}\right)\cap D_j\right|}{\left|p\left(out_j^{Description}\right)\cup D_j\right|}\right) \times Rank\left(out_j^{Description}, D_j\right)\right]/J \qquad (7.16)$$

$$sim_{Density}\left(OUT_{MP_j}, R_{MC}\right) = \left[\sum_{j=0}^{J}\left(\frac{N\left(out_j^{Description}, D_j\right)}{d\left(out_j^{Description}, D_j\right)}\right) \times Rank\left(out_j^{Description}, D_j\right)\right]/J \qquad (7.17)$$

Utilizing the above modifications can result in the removal of unneeded service description properties/concepts. Consequently the semantic similarity of two service ports can be

evaluated in a shorter periods of time and with greater levels of accuracy. This process is now performed to determine the semantic validity of the MP according to the SSON composition plan. An acceptable level of semantic similarity along with an acceptable level of QoS places the MP as one of the possible next-hop candidates. The current MP then ranks all received SSON join acceptances and determines the highest ranked MP. That node is then contacted to finalize the next-hop composition. SSON path formation then continues to find the next hop, the one after, and so on until the MC is reached.

7.5.2. Handling Query Failures in Plan-Free SSON Compositions

The semantic nearness approach for MP query in Section 7.3 can only be applied if the query can successfully return at least one MP whose output port nearness level to the MC's requirements is greater than that of the previous MP. In the event a query fails to successfully find an adequate MP, we propose the use of an alternate scheme. Figure 7.2 summarizes the required steps.

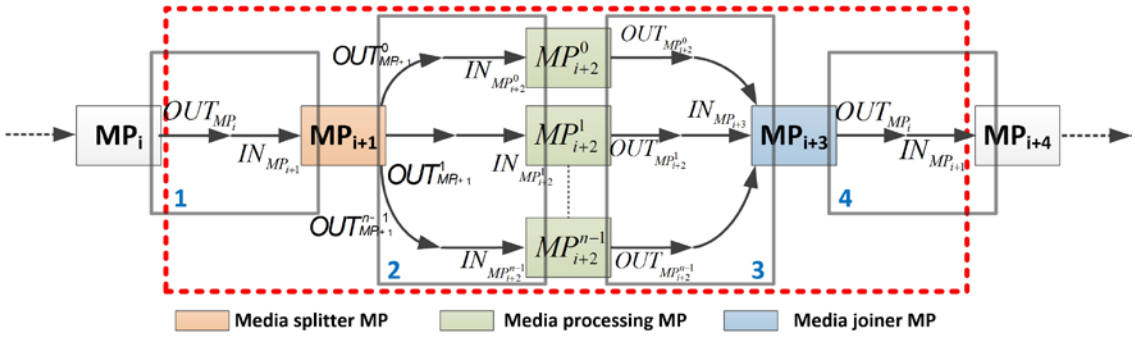


Figure 7.2 A visual representation of steps required to overcome MP query failures.

To successfully overcome a MP query failure four steps are needed:

1. Search for a media splitter service, MP_{i+1}. The splitter takes a single media stream input and splits it into *n* output streams.
2. Search for *n* services; one for each of the split streams, $MP_{i+2}^{0}, MP_{i+2}^{1}, \ldots, MP_{i+2}^{n-1}$. This is performed using the semantic *nearness* evaluation method.
3. Search for a media joiner service, MP_{i+3}. The joiner takes *n* media stream inputs and

merges them into a single output stream.

4. Continue querying for next-hop MPs using methods of Chapters 5, 6 and 7.

If finding a single service capable of producing an output stream that is semantically closer to the MC requirements than the current MP has failed, the first step is to find a media splitter MP that can break down the media stream into multiple streams. The splitter's input port should semantically match the output of the current MP (semantic similarity, Section 7.3). The input port's and output streams' QoS should also remain within the MC's acceptable range (fuzzy-based QoS evaluation, Section 6.3). If a MP that meets these requirements is found then the process moves to the second step.

In the second step, queries are sent to search for MPs capable of providing services that take as input one or more of the split media streams. For a MP to be accepted, its service must: 1) have an input port that meets the semantic similarity conditions of Section 7.2 when compared to the respective output stream from the splitter, 2) have input and output QoS levels that remain within the MC's acceptance range, and 3) have an output port that generates a stream that is semantically nearer to the MC's requirements than that of the previous MP, Section 7.3. For this process to be successful, at least one of the split media streams should pass through a MP that brings it, i.e. that particular stream, semantically closer to the MC's requirements.

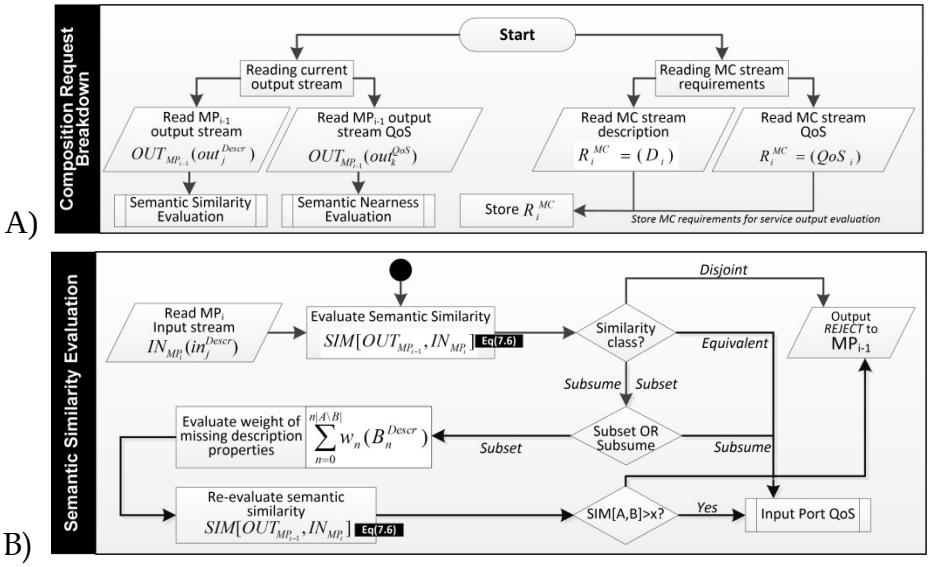
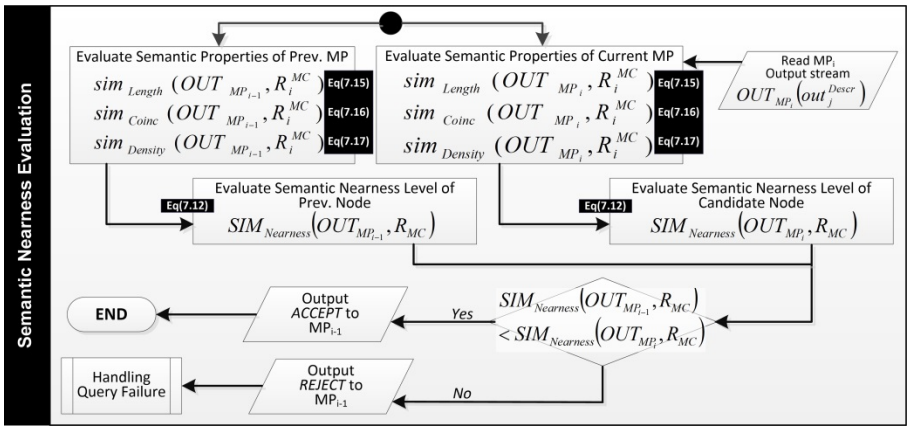
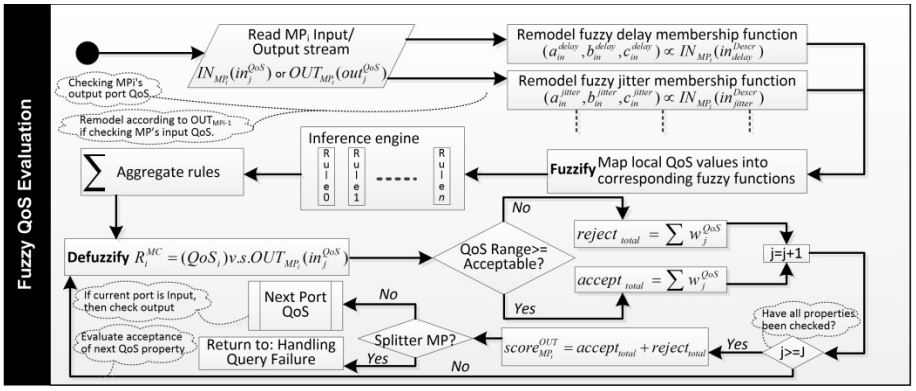


Figure 7.3i **A flow chart summarizing all steps necessary to perform the plan-free semantic nearness SSON Composition. The chart shows the first two major components: A) Composition Request, B) Semantic Similarity Evaluation.**

C)



D)



E)

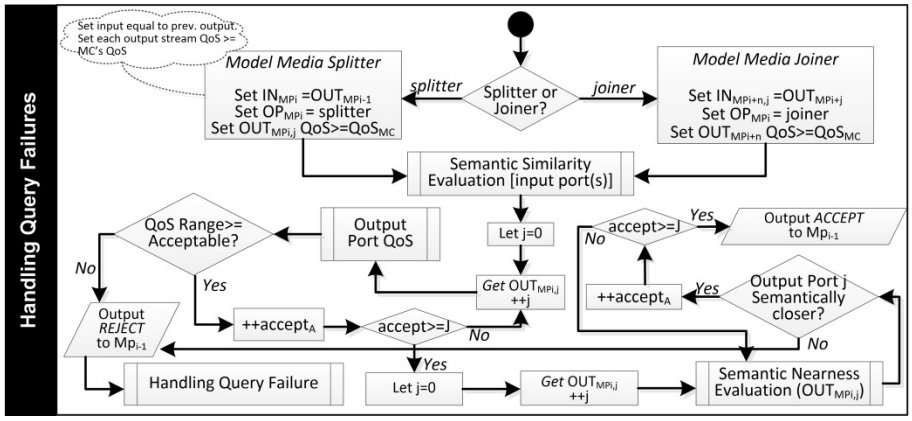


Figure 7.3ii A flow chart summarizing all steps necessary to perform the plan-free semantic nearness SSON Composition. The chart shows the last three major components: C) Semantic Nearness Evaluation, D) Fuzzy QoS Evaluation, and E) Handling Query Failures.

The final decision regarding the media processing MPs, $(MP_{i+2}^{0}, MP_{i+2}^{1}, \ldots, MP_{i+2}^{n-1})$, rests with the splitter MP. It is also responsible for querying for the joiner MP in step 3. The joiner must accept as input all *n* media streams; those that have been modified and those that have not. This is based on the semantic similarity of Section 7.2. Each input port of the joiner must match one of the incoming media streams. In the final step the joiner MP

searches normally for the next-hop MP. The process simply follows the normal MP search method presented in this thesis. Figures 7.3i and 7.3ii summarize all steps necessary to perform a successful SSON composition. The figure illustrates the semantic similarity, semantic nearness, and fuzzy-based QoS evaluations as well as how query failures are handled. The figure if broken down into five major sections: Composition Request Breakdown, Semantic Similarity Evaluation, Semantic Nearness Evaluation, Fuzzy QoS Evaluation, and Handling Query Failure. The semantic nearness component is further broken down into three subsections for semantic length, coincidence, and density. Moreover, the QoS evaluation is divided into two sections, one pertaining to QoS of the input port while the other for the output port. All details of the above components have been discussed in this chapter as well as Chapters 5 and 6.

7.6. System Evaluation

7.6.1. Simulation Setup

The work presented in this chapter focuses on the mechanism of choosing appropriate MP nodes to successfully compose SSONs that meet MC quantitative and qualitative requirements in mobile environments without reliance on composition plans or workflows. As such we have evaluated the performance of our solution by testing it on a number of MP discovery mechanisms. The goal is to illustrate that the cost of incorporating our semantic-based MP comparison process into existing discovery mechanisms is within acceptable levels. We also compare the semantic-nearness composition method of this chapter with the plan-based H-SON composition approach of Chapter 6.

Our experiments are conducted using the C++-based OMNET++ [111] discrete event network simulator. The simulator permits the design of network models through a hierarchical architecture of system modules and sub modules. Topologies including the hybrid SON are built over the open-source overlay and P2P network simulator OverSim [112] for OMNET++. OverSim is characterized by a graphical interface illustrating the overlay and underlay topologies and network packets exchanged between modules. Statistics are collected by a central module that processes statistics information.

Nodes are placed in a 2-dimensional Euclidean space where delay between any two nodes is proportional to the distance between them. Nodes' arrivals and departures are based on a churn model that uses lifetime exponential distribution. To model the behavior of the network under realistic traffic conditions a fully-recursive KBR protocol is implemented. A sample application program of simple request-response encapsulated messages runs over the network and forwards messages between nodes according to local routing tables. Message sizes range from 32 bytes to 100 bytes. Hybrid SON Join messages are sent at 10-second intervals with a maximum of two retries. Stabilization messages are set with 120 seconds, 5 seconds, and 20 seconds intervals for Chord finger reconfiguration, predecessor update, and stabilization messages respectively. Modifications implemented to the normal Chord protocol are further discussed in Chapter 4. The simulations involve up to 100 nodes randomly distributed and divided into 5% MCs, 5% MSs, and 90% MPs. Each MP is associated randomly with a set of services where each service is provided at specific QoS levels. Service and QoS descriptions are specified in OWL/RDF ontology files. MS nodes are associated randomly with a set of resource and media description ontology-based files. Additionally, MCs are associated with media request descriptive files that list the media types and QoS levels to be requested from MSs. Nodes are added to the network according to a Poisson process at a rate of 1 node/second. To closely model the dynamicity of mobile networks, node joins and failures are modeled through a Poisson process with a mean arrival rate of 30 seconds.

Semantic nearness evaluations require formal ontology-based service descriptions. OntoCAT [197], an open source java-based library is used to parse, search through, and compare acquired service description files. OntoCAT provides a high level abstraction for interacting with ontology resources in the standard OWL format. It provides a seamless programming interface to query heterogeneous ontology resources where each resource is wrapped behind Java service commands. OntoCAT exists in a number of flavors including stand-alone database and browser, REST services, and OntoCAT R package. Furthermore, it provides a robust, configurable solution for accessing ontology terms specified in local files. We use OntoCAT to perform the semantic nearness evaluation discussed in Section 6.1.

7.6.2. Testing Over H-SON and Limited Broadcast

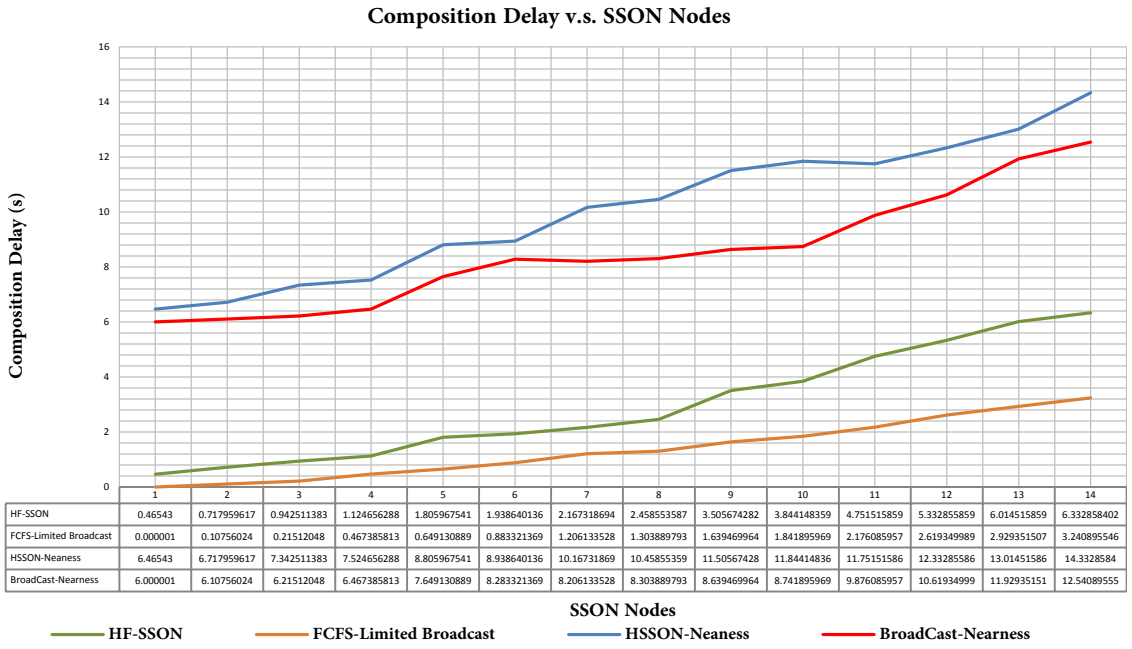
Earlier in Chapter 4 we introduced a hybrid SON structure (H-SON) that consisted of multileveled Chord ring structures that simplified MP discovery based on service type and quality levels. Through a H-SON, the advantages of both structured and unstructured overlays were maximized while minimizing their disadvantages. The hybrid structure was characterized by its high level of dynamicity due to the dynamic promotion and demotion of nodes based on their stability levels, their services' quality levels, and the popularity of their provided services.

We showed in Section 4.5 how, when compared to normal Chord-based overlay networks, the presented H-SON experienced significantly better performance. The average delay experienced by each node attempting to join the network in the H-SON was approximately 30-50% lower than that of a normal Chord ring with the same number of nodes. Stabilization messages needed to maintain links in mobile networks especially in structured overlays such as Chord rings increase with the number of joined nodes increases. The increase stems from the large exchange of messages to maintain links to the successor, predecessor, and finger links. On the other hand, in the H-SON the reduced number of nodes present in each Chord ring and large number of leaf nodes is reflected through a decrease in the number of stabilization messages exchanged. The H-SON registered a reduction of approximately 62% in rate of maintenance messages compared to normal Chord structure.

A plan-based SSON composition method was presented in Chapter 5 built over our hybrid SON underlay. The plan included abstract ontology-based models of every MP required in the SSON path. Consequently, only a semantic similarity evaluation process (Section 7.2) was required to compare the candidate service's operations, input and output ports to the plan descriptions. Evaluating semantic nearness was not needed. In our first evaluation, we tested the composition performance of the plan based solution and compared it with our plan-free composition under the condition that queries in the plan-free method were only directed towards the same hybrid SON regions the plan-based method queried.

The same test was applied to a limited-broadcast (LB) MP discovery method over the H-SON. Two iterations of LB are simulated. The first involves nodes that forward query

messages to *n* neighbors (*n=7* in this case). MPs blindly broadcast queries to *n* direct neighbors in an attempt to find suitable next-hop MPs. Forwarding is limited to 4 hops, after which the query is dropped. The first MP to reply with a composition acceptance is chosen as a next-hop node. This approach followed a MS-generated plan with which discovered MPs were compared. The second iteration of LB was plan-free. Discovered MPs performed the four semantic and QoS evaluation steps to determine whether a composition was moving semantically closer to- or further away from the MC's goals.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14
HF-SSON	0.46543	0.717959617	0.942511383	1.124656288	1.805967541	1.938640136	2.167318694	2.458553587	3.505674282	3.844148359	4.751515859	5.332855859	6.014515859	6.332858402
FCFS-Limited Broadcast	0.000001	0.10756024	0.21512048	0.467385813	0.649130889	0.883321369	1.206133528	1.303889793	1.639469964	1.841895969	2.176085957	2.619349989	2.929351507	3.240895546
HSSON-Neaness	6.46543	6.717959617	7.342511383	7.524656288	8.805967541	8.938640136	10.16731869	10.45855359	11.50567428	11.84414836	11.75151586	12.33285586	13.01451586	14.3328584
BroadCast-Nearness	6.000001	6.10756024	6.21512048	6.467385813	7.649130889	8.283321369	8.206133528	8.303889793	8.639469964	8.741895969	9.876085957	10.61934999	11.92935151	12.54089555

Figure 7.4 **Experienced composition delay over the H-SON and LB using plan-based and plan-free semantic nearness evaluation methods.**

Figure 7.4 compares the SSON composition delay in four cases all running on the hybrid SON. The first two are the plan-based and LB. The former involves fuzzy-based QoS evaluation and semantic similarity to query MPs. The latter only involves a simplified semantic similarity evaluation to find MPs that match plan-based tasks. The other two cases are the plan-free and LB while using semantic similarity, semantic nearness, and fuzzy-based QoS evaluation.

Results illustrate how delay increases steadily with the number of MPs involved in the SSON path. The plan-based delay is slightly higher than that of LB. This increase is due to

LB's lack of fuzzy QoS evaluation and its simplified semantic comparison. LB thus composes SSONs faster however it lacks semantic and QoS precision. Furthermore, the flooding approach of LB results in a number of messages exchanged that far exceeds the plan-based method where messages are only forwarded to specific overlay sector bands.

Performing the same two tests while using semantic similarity/nearness and fuzzy QoS evaluations presented earlier, an expected increase in composition delay is seen. On average the increase is in the range of 5.8-6.3 seconds. However, the benefits of enhancing MP service discovery and match-making through semantic nearness evaluation greatly outweigh the costs.

7.6.3. Testing Using SORD Service Discovery

Al-Oqily et al. [32] proposed a service composition algorithm that deals with decentralized, dynamic, and seamless composition of multiple autonomic elements into service specific overlay networks. The algorithm combined service discovery and composition into one step. The algorithm is built on and extends the SMART architecture [14]. The presented work makes an important assumption. It assumes each MP has a distance function that is used to produce the list of required adaptations for a media flow based on its input and output. However, the authors do not present a methodology for quantifying the compatibility and similarity between service descriptions. An assumption is made that there exists a function $sim(MD_1, MD_2)$ that computes the difference between two media descriptions. The function also represents the set of adaptations that are required for a media flow to be viewable by the MC, and computes the similarity between MPs. A function introduced by Herborn et al. [137] is assumed to be available. However the similarity function relied on a globally accessible directory of MP descriptions. A distributed directory over a Chord structure based on UDDI was utilized. Such approach presents scalability and stability issues in highly mobile environments.

Our solution overcomes these scalability problems by providing a decentralized MP similarity evaluation mechanism. Comparing our work in this chapter to [32] is inconsequential. This is because the authors' aim was to provide an efficient MP discovery

mechanism while our work focuses on the means of evaluating the similarity between MPs and using this information to dynamically and automatically compose services while meeting MC QoS requirements regardless of the MP discovery mechanism used. In [32] the authors evaluated the efficiency of their SSON composition algorithm. However it was not clearly illustrated how MPs were compared after discovery according to their proposed direction-based discovery mechanism. Furthermore, an overly simplified list of services was used. The authors considered a service model that transformed one alphabet to another (for example, a service can accept *a* as input and transforms it to *b*, *a*→*c*,…, *b*→*a*,…,*etc*.) resulting in a total of 625 different services.

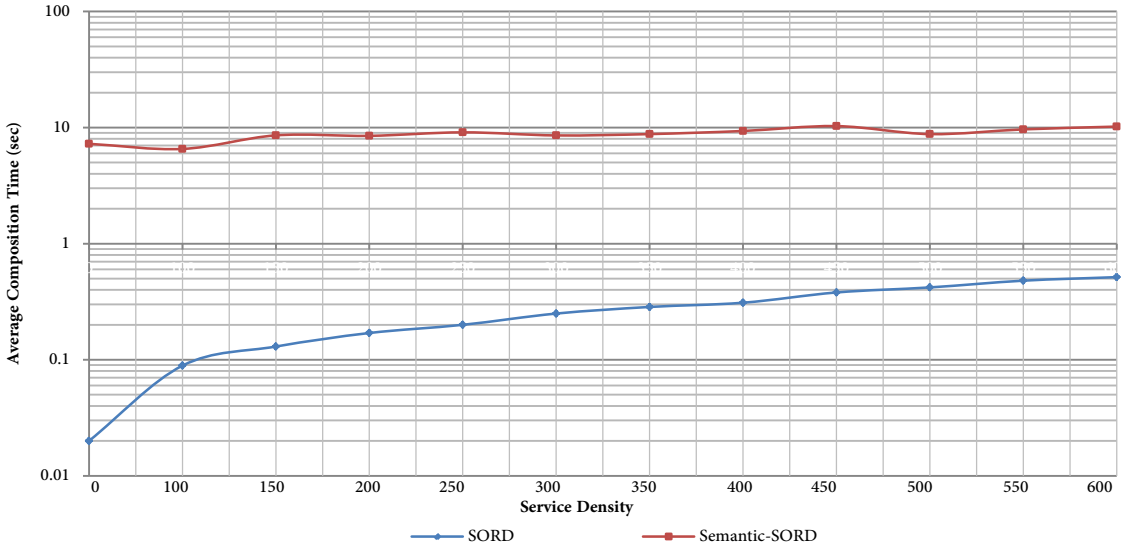


Figure 7.5 **Average SSON composition delay comparison between simulation results as presented in [32] and our enhanced fuzzy QoS and semantic nearness evaluation MP comparison method.**

We have duplicated the simulation setup presented in [32] but with the use of OMNET++ and OverSim rather than the BRITE [198] topology generator. The topology had 2000 nodes in a 1000x1000 node two-dimensional overlay space. We ran the simulation 13 times with varying values for the search angle α (5-45°). However, the simplified service descriptions have been replaced with ontology-based media descriptive files with random QoS properties' levels used in our previous simulations. Furthermore we used our semantic similarity,

semantic nearness, and fuzzy-based QoS measurements to determine whether a MP service is capable of joining an SSON composition. We then compared our results to those presented in [32].

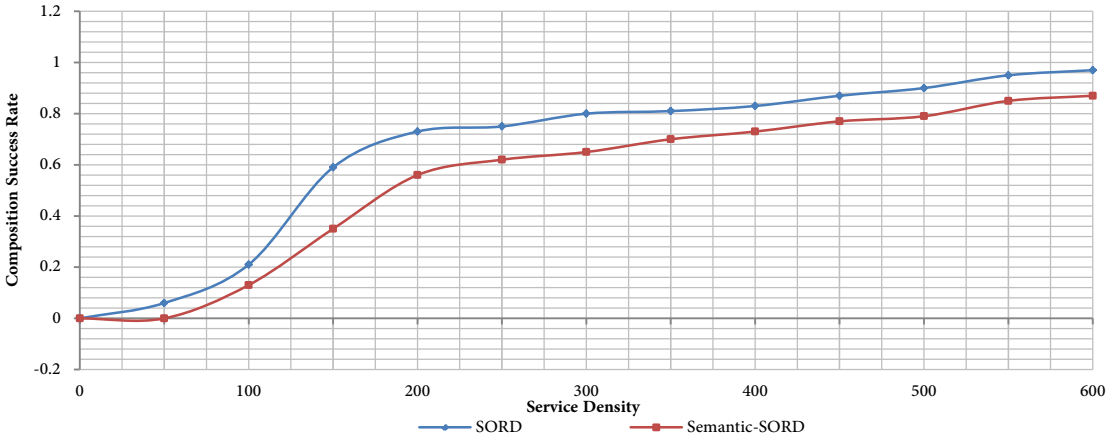


Figure 7.6 **Composition success rate of our MP semantic evaluation method compared to results presented in [29].**

In Figure 7.5 the average composition time is the difference between the start of the composition request and the arrival of the completed SSON path. The original results indicated average composition times ranging from 0.1 to 0.9 seconds. New simulation tests involving semantic-based evaluation and fuzzy-QoS comparisons indicate an increase in the total time required to compose SSONs that range from 8.5 to 10 seconds; an 8 fold increase. However, this can be explained by our use of realistic and more complex service description models, as well as utilization of syntactic, semantic, and fuzzy QoS comparisons when selecting MPs in the new simulations. The original simulations in [32] on the other hand did not take the above into consideration.

Furthermore, Figure 7.6 illustrates how the composition success rate has not been negatively affected by our algorithm. Success rate is defined as the number of requests that receive positive responses divided by the total number of queries. Although success rates in our simulations slightly fall below those of the original SORD, yet they can be justified by our increased accuracy of $sim(MD_1, MD_2)$ functions compared to those used originally in [32]. Our composed SSONs better meet the MC's media flow format and QoS requirements.

Moreover our simulations assess the performance of SSON composition with more realistic service models than those presented in the original SORD simulations.

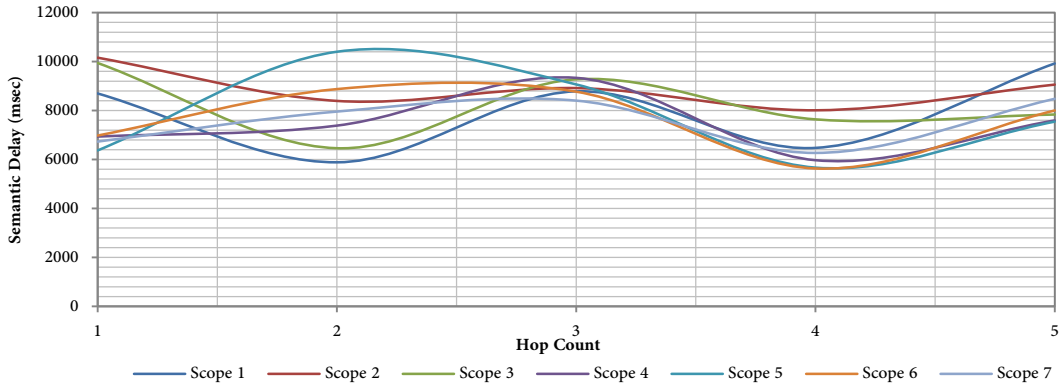


Figure 7.7 **Semantic delay experienced by the system with varying hop counts and stable search scopes at seven different levels.**

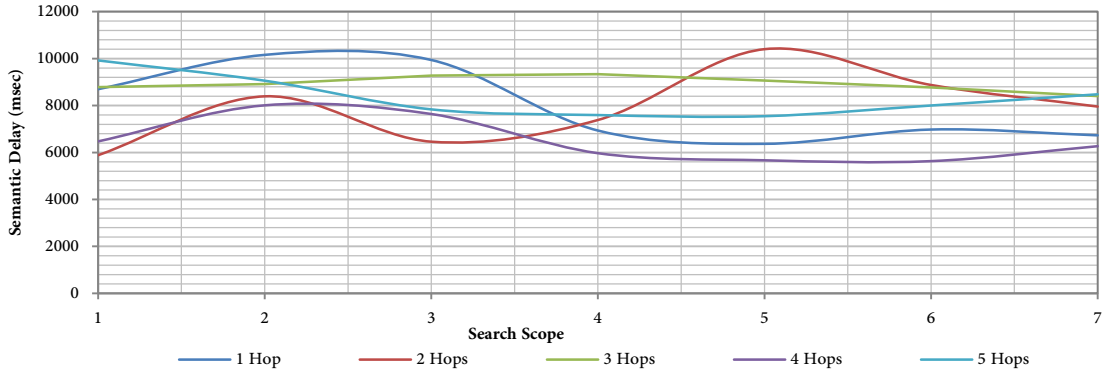


Figure 7.8 **Semantic delay experienced by the system with varying search scopes and stable hop counts at five different levels.**

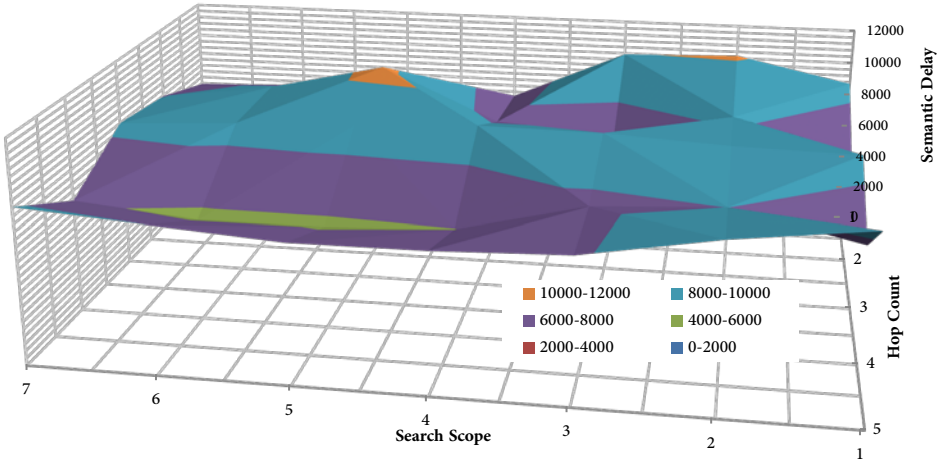


Figure 7.9- Semantic delay for SSON composition as hop count and search scope values vary. The figure displays a high level of stability.

7.6.4. Semantic Nearness Stability Evaluation

An important metric to illustrate the viability of our SSON composition algorithm is its stability level. Stability is defined as the system's bounded range of expected delay such that under most circumstances no unexpected deviations in terms of delay are experienced by the system. Figure 7.7 shows how semantic nearness evaluation delay levels vary with changing hop counts given various static search scopes. The same simulations are repeated in Figure 7.8 with dynamic search scopes while holding hop counts at five static levels. Figure 7.9 merges hop count and search scope to illustrate the range of stability under the variation of both metrics. The results clearly show that delay experienced by each node during SSON composition pertaining to the evaluation of its output media flow's semantic nearness to that of the MC's requirements, and comparing this value to the previous node's output flow's semantic nearness to the MC's requirements is a bounded value. The delay falls between 5 and 10 seconds with uniform changes in delay levels. The results clearly show a system with high levels of stability. Consequently our semantic nearness-based solution can be trusted to behave in an expected and stable nature.

7.6.5. Semantic Nearness Enhanced H-SON and Limited Broadcast Performance Evaluation

As noted earlier, the behavior and usage feasibility of our candidate MP selection algorithm is affected differently depending on the underlying MP discovery method used. We perform a number of simulation tests to show the behavior of the system when used over the H-SON and when used over LB.

Figures 7.10 and 7.11 show the average total number of composition paths generated from a single SSON composition request for LB and H-SON respectively. A fixed MP density of 100 nodes was used. Larger numbers of chosen paths reflect the flexibility of the system and its adaptability to choose an SSON path that best meets the MC's requirements. Moreover, larger numbers of paths reflect a higher probability of successfully forming SSON compositions in networks with lower MP densities. Using limited broadcast, Figure 7.10, the total number of composition paths found remains close to 0 for low search scope and hop

count values. This is expected since the chances are very low of successfully finding all needed MPs in a multi-MP SSON path within immediate neighboring nodes. As the query is sent deeper into the network and to a larger number of MPs, the number of composed paths greatly increases.

Given the advantages of the H-SON as discussed earlier, it is expected that our semantic nearness MP candidate identification method will perform better than LB. Figure 7.11 confirms this conclusion. Greater numbers of SSON paths are composed under lower hop counts and search scopes. The reason stems from the fact queries can be directed towards regions of the hybrid overlay where it is more likely MPs with favorable services and QoS levels are found.

From the above simulation tests, a measurement pertaining to query success rates can be collected. Success rate is defined as the number of requests that receive a positive response divided by the total number of queries sent over the network. Figure 7.12 shows that query success rates in the H-SON can reach a range of 90-99% when a balance exists

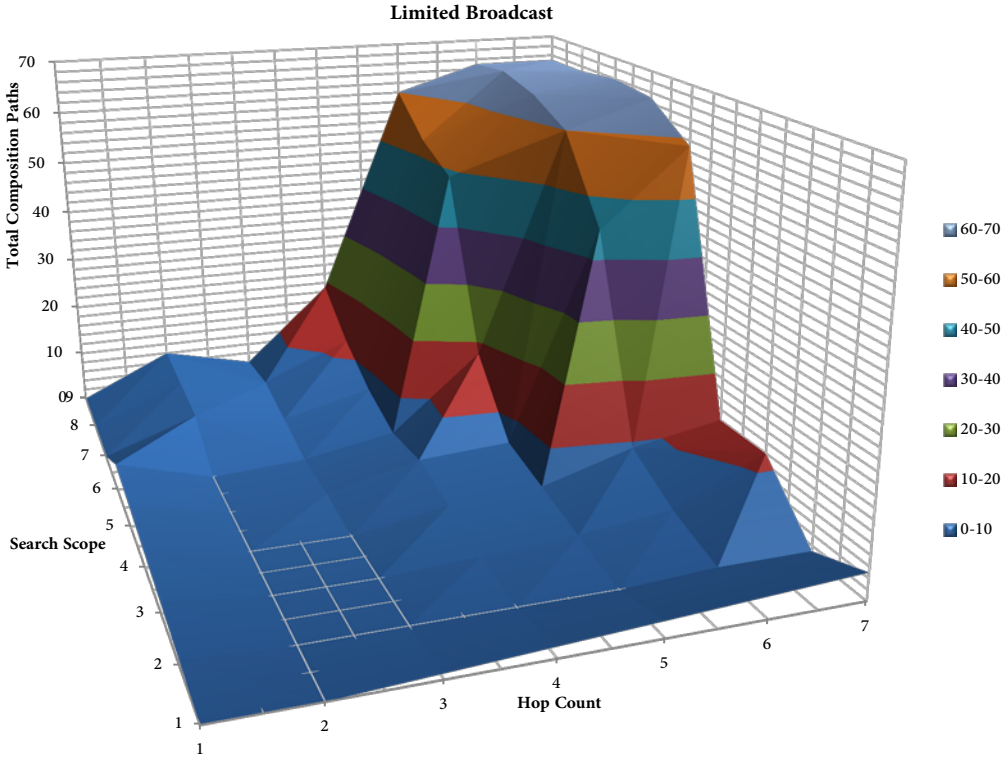


Figure 7.10 **Total composition paths generated in LB discovery method with varying search scopes and hop counts.**

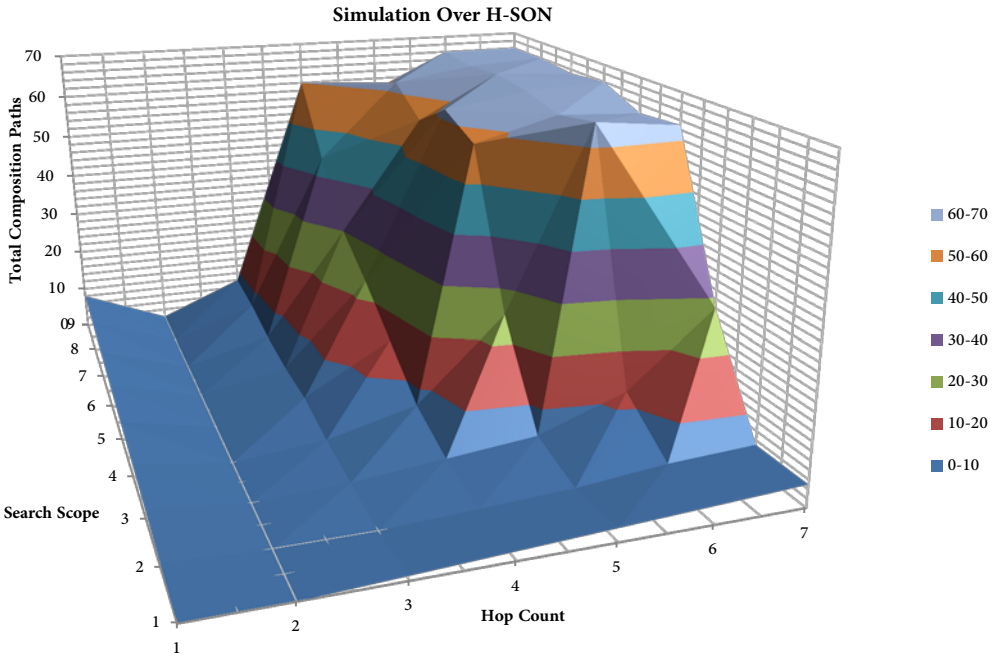


Figure 7.11 **Total composition paths generated in H-SON with varying search scopes and hop counts.**

between the number of hops and query scope. With low search scopes, high hop counts result in higher success rates. With high search scopes, low hop counts result in higher success rates. This can be explained by the fact most MPs with appropriate QoS levels and service descriptions can be reached in a single or at most a few hops. A sharp drop in success rate is seen at higher hop counts and scopes. That is due to the majority of excess messages reaching candidate MPs with the initial phase of queries while the remaining queries reach nodes whose potential for joining the SSON composition are slim. Consequently the usefulness of delivered messages decreases and the number of join rejections increases thus increasing failure rates.

The same experiment was performed using LB MP discovery method (Figure 7.15). Contrary to the H-SON, LB experienced higher message success rates at lower hop counts and intermediate search scopes. The random distribution of MP services in the network indicates it is more likely to successfully find needed MP services in the nearby vicinity of the current hop MP, if a wider area is queried and if the query is sent deeper into the

network. Decreases in message success rate at the low- and high-end spectrum of the search scope illustrates two important characteristics of LB SSON composition with semantic nearness and fuzzy QoS evaluations. First, it shows how forwarding messages to only a limited number of immediate neighbors reduce the probability of receiving a positive query reply. Second, it shows that sending messages to large numbers of immediate neighbors increases the number of rejected messages. This is caused by the larger numbers of MPs incapable of providing necessary services are contacted and thus send back their rejections. Figures 7.13 and 7.14 illustrate the behavior of the message success rate for H-SON using the plan-free semantic nearness composition approach by varying hop counts and search scopes respectively. Through such information we can find the best approach to maximizing query success rates while minimizing the number of exchanged messages. As illustrated in Figures 7.13 and 7.14, query success rates increase with large search scopes that keep the hop counts to a minimum or small search scopes that search deeper into the network through increased hop counts. Similarly, Figures 7.16 and 7.17 illustrate the same information for the Limited Broadcast search method enhanced with semantic nearness evaluations. However in Figures 7.16 and 7.17 the success rate is highest when the hop count is kept at a minimum for large search scopes given the greater possibility of query failures as the ratio of discovered MP services compared to sent queries continues to decrease.

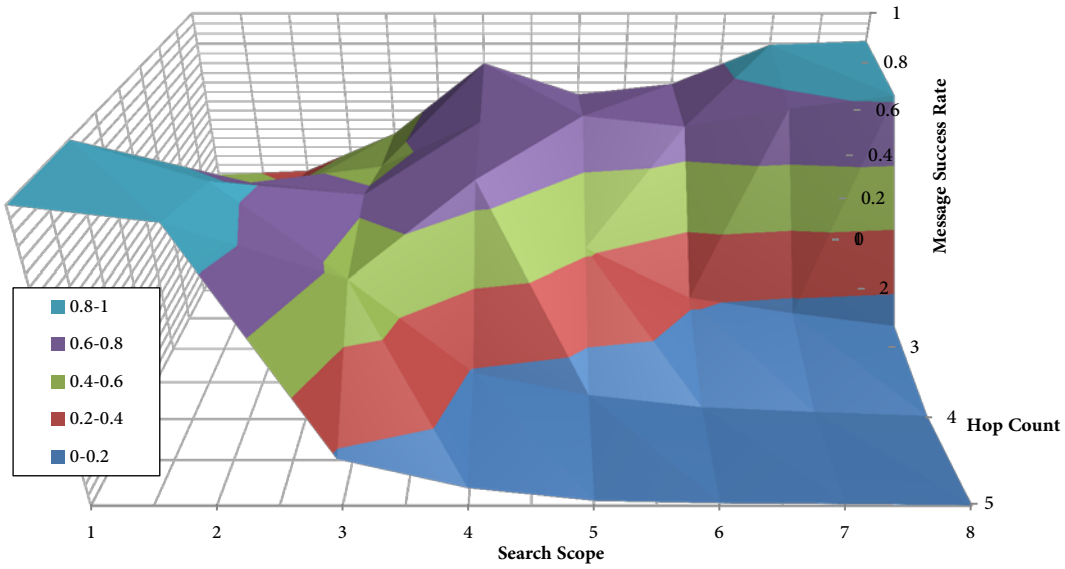


Figure 7.12 Message success rate with varying hop counts and search scopes for H-SON.

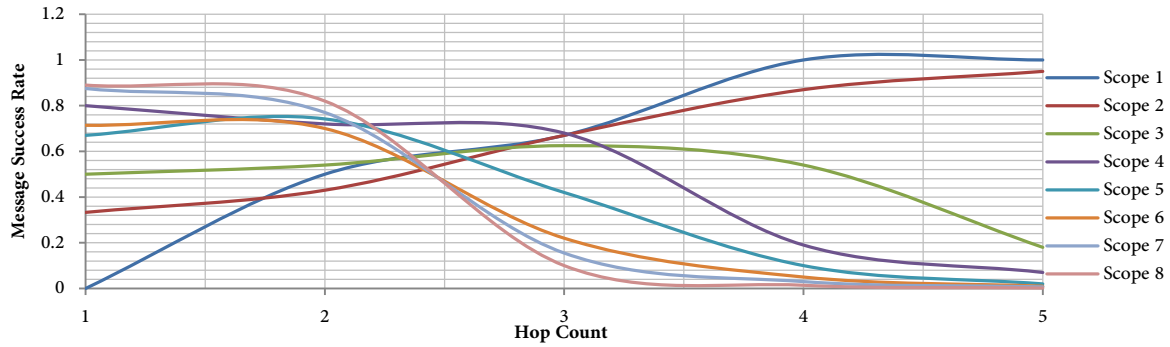


Figure 7.13 **Message success rate with varying hop counts and static search scopes for H-SON.**

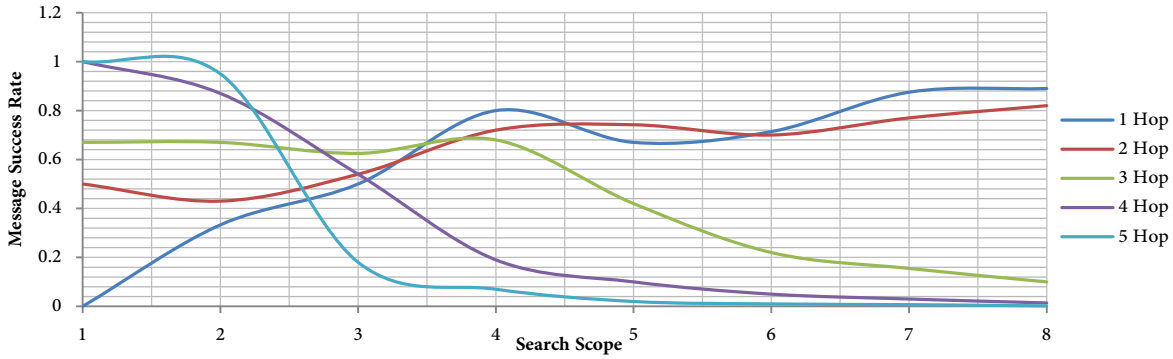


Figure 7.14 **Message success rate with varying search scopes and stable hop counts in the H-SON. The test is performed for five variations of hop counts.**

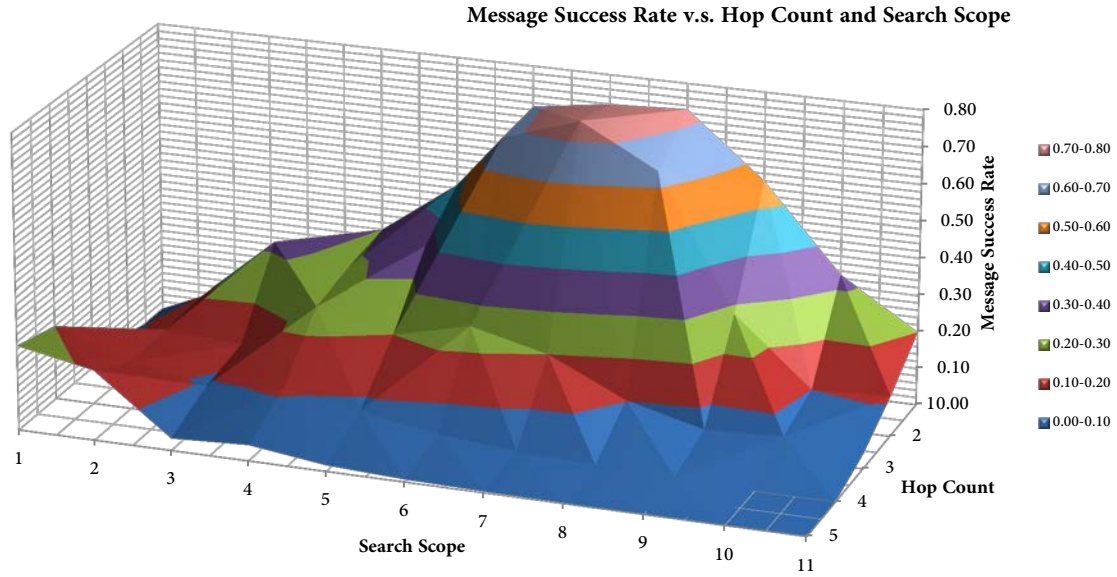


Figure 7.15 **Message success rate with varying hop counts and search scopes for LB service discovery mechanism.**

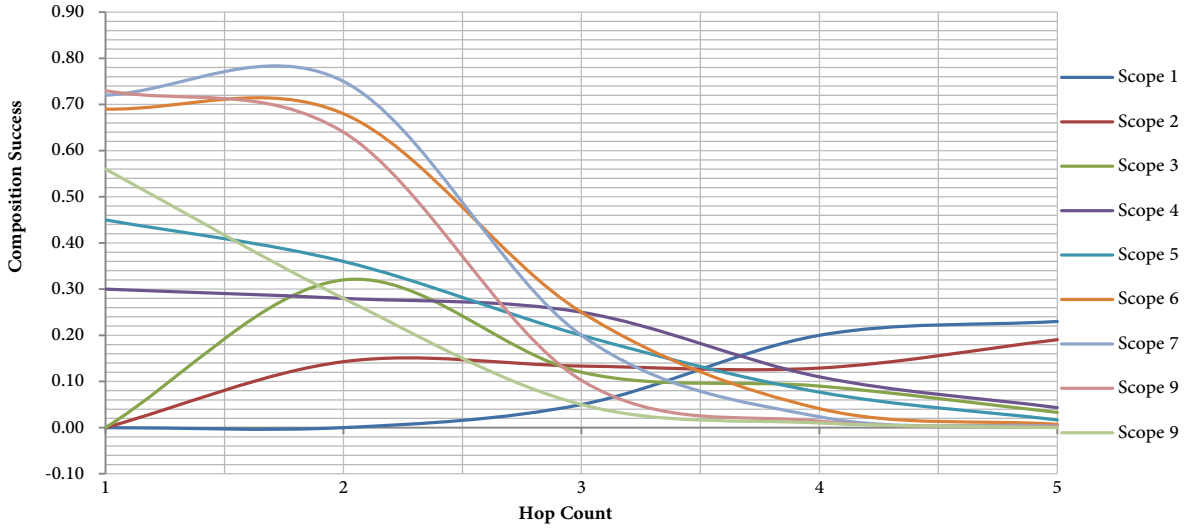


Figure 7.16 **Message success rate with varying hop counts and static search scopes for the LB service discovery mechanism. Test is repeated for 9 different search scope variations.**

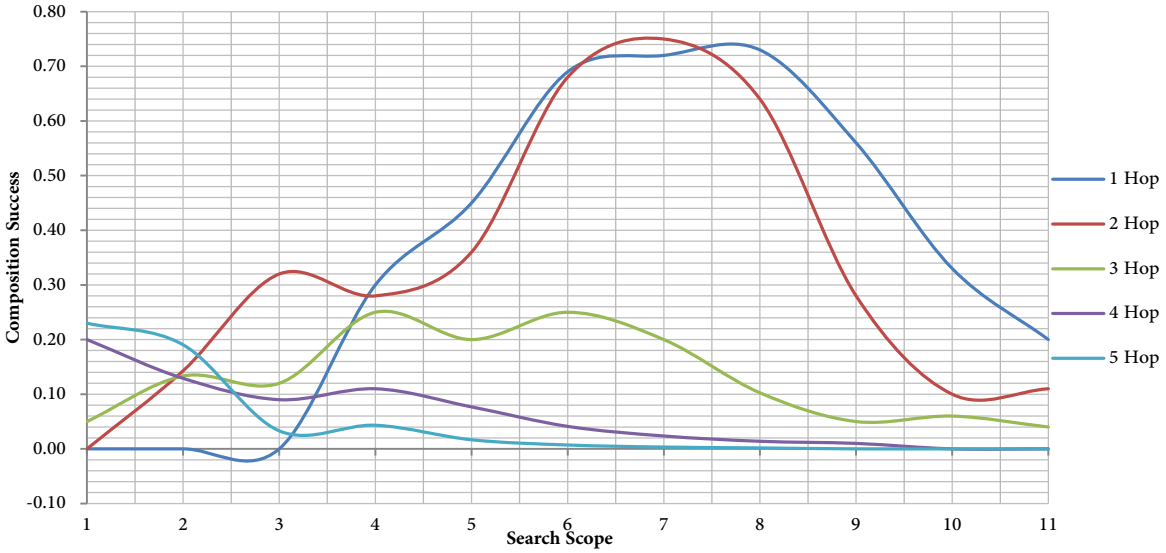


Figure 7.17 **Message success rate with varying search scopes and static hop counts for the LB service discovery mechanism. Test is repeated for 5 different hop count variations.**

7.7. Summary

This chapter presented two steps of our semantic service evaluations used in the plan-based and plan-free compositions. An extension to a semantic similarity evaluation of MP

ports and functionalities provided in the literature increased accuracy for MP service selection in SSON compositions. Similarity was classified into four categories depending on the number and type of ontological properties that match between discovered services and plan-based tasks. Similarity was used to rank discovered services. The highest ranked MP with the QoS level that best matches the MC's requirements was chosen as part of the compositions.

Furthermore, this chapter presented a plan-free, semantics-based dynamic service composition solution for mobile environments. The system provides a decentralized solution that dynamically composes services based on MC media flow and QoS requirements. Moreover, the chapter presents a 4-step solution that utilizes fuzzy-based QoS evaluation that can match the QoS levels of interlinked MPs while maintaining the acceptable QoS range of the MC. Additionally, a semantic nearness evaluation process was introduced thus presenting MPs with the ability to automatically and independently evaluate the feasibility of a service composition based on the semantic gain acquired from choosing one MP service over another. The process guarantees a solution that semantically advances towards the MC's requirements with each SSON hop.

The empirical evaluation of the system presented in the chapter provides insight to the behavior of the proposed SSON composition solution. Results showed that our solution is characterized by high stability and could be implemented over any existing service discovery mechanisms. Moreover, the costs associated with the fuzzy QoS, semantic similarity, and semantic nearness evaluations remain within acceptable levels for the benefits gained in terms of accuracy to the MC's media flow description and QoS requirements.

Chapter 8

Conclusion and Future Research Directions

This chapter outlines the contributed research work and discusses planned and directions for future work. Section 8.1 summarizes the conducted research and the main contributions in the area of autonomic QoS-based SSON composition in mobile networks. Section 8.2 sheds a light on the planned future research directions. Section 8.3 presents some of the limitations of the work presented in this dissertation and methods to apply in the future to improve its functionality.

8.1. Conducted Research Work

The focus of the conducted research has been the development of an autonomic and automatic SSON composition method in dynamic networks. The first step towards achieving that goal included a literature study of the service composition problem and the difficulties associated with mobility and guaranteeing user satisfaction in terms of QoS. We aimed to address two questions: 1) what are the requirements of SSON composition? 2) why current composition approaches do not satisfy these requirements? Answering these questions required a prolonged survey of major research directions and efforts made over the last decade in the areas of autonomous overlay network management, service discovery schemes and composition, as well as semantic and syntactic based service comparison methods. Based on the identified limitations of current research work, a novel framework for an automated SSON composition system has been designed. The framework has been presented as a multilayered model that automatically adapts to the dynamicity of mobile

devices. In the following, a summary of the main contributions of current research work is given:

- A complete breakdown of the autonomic composition problem into a three-layered architecture. Focus was directed at the two topmost layers; the SON and SSON. Moreover, a three-part solution was conceived utilizing a unique hybrid SON layer, a fuzzy logic-based QoS evaluation MP discovery method.
- Design of a novel hybrid SON underlay. It is based on a hierarchical organization of structured and unstructured topologies involving MPs, MSs, and MCs. The semantics of MP operations, the QoS levels of provided services, and the stability of involved nodes were used to achieve the highest possible level of dynamicity and performance. In contrast to existing methods, the proposed H-SON enhanced the MP discovery process by clustering nodes according to provided services and the quality of such services. The presented topology produced a reasonable message overhead, and simulation results had shown the efficiency of the presented scheme.
- A novel scheme for QoS-based MP discovery has been presented. It is based on a widely known family of fuzzy logic membership functions known as the bell-shaped curve. In contrast to existing fuzzy-based composition approaches, our design utilizes the H-SON, MC requirements, and MS-generated composition plan to adapt and provide the most accurate MP discovery solution. The scheme is applied internally in every node, to pinpoint potential next-hop MP regions in H-SON, and to rank candidate next-hop MPs in the SSON. Simulation results illustrated the low cost associated with utilizing a fuzzy-logic system and the enhanced accuracy and performance of MP discovery compared to other schemes.
- A plan-free SSON composition mechanism that utilizes semantic and syntactic nearness evaluation methods. The composition mechanism is completely decentralized and does not required a MS-generated plan and does not rely on the H-SON we presented as an aid in one of the possible service composition solutions. The semantic nearness evaluation process presented MPs with the ability to perform service composition feasibility tests automatically and independently. Composition decisions were made based on the semantic gains acquired from choosing one MP

service over another. Consequently, with each MP service chosen at every SSON hop, our solution guaranteed that composition advanced towards the MC's media flow and QoS requirements.

8.2. Future Research Work

The main focus of our future research work can be divided into three key directions as follows: QoE-based SSON composition and adaptation, Learning-based adaptive service composition and reconfiguration, and proactive adaptation and seamless handover of SSONs as mobile users move from one network technology to another.

8.2.1. QoE-based SSON Composition and Adaptation

There are several situations that require service composition adaptations. These include changes or emergence of new requirements, changes in the context of participating service nodes, changes in the quality levels of such services, and the arrival or failure of MPs. Guaranteeing specific levels of QoS in transmitted media streams such as delay, availability, response time, jitter, and packet loss, is usually challenging to transpose to the experienced level of quality observed by users. The quality of experience (QoE) of a transmitted media stream can be deemed inadequate even with fast response times, low jitter, and packet loss rates. The converse is also true. As such any SSON composition mechanism should take the users' QoE into consideration to guarantee the highest level of user satisfaction. QoE considers both the services' and the users' quality influencing factors [199] [200] [201]. QoE properties include emotions, motivation, experience, and goals. Moreover, the QoE measure has a unique meaning according to the specificity of the transmitted media stream or provided service. The most significant factor affecting QoE in media stream flows is delay variations in network transmission. This is referred to as the network delay jitter. Some mechanisms exist to overcome jitter variations. One possibility includes the integration of some QoS mechanisms within the network to deal with the specific needs of some of the existing delay-sensitive services through policy enforcement and adaptation

[202]. Another approach relies on deployment of control and adaptation mechanisms at the sender and receiver sides. Through this approach MSs can control their transmission rates to obscure quality degradations experienced in the SSON path. Additionally, MCs can absorb and hide delay jitters through utilization of adaptive jitter buffers.

8.2.2. Learning-based Adaptive Service Composition and Reconfiguration

Empowering service discovery and composition mechanisms with hierarchical reinforcement learning provides it with high levels of adaptability. Reliance on SSON monitoring methods imposes intensive work to monitor, tune, and reconfigure composition paths as network conditions fluctuate. Consequently, it is highly desirable that service composition be self-adaptive in dynamic environments to reduce maintenance costs. Reinforcement Learning is a typical technology used for planning and optimization in dynamic environments. Learning aims at finding the optimal policy to deliver the best quality of service as changes in environmental conditions occur. Although the service composition mechanism presented in our work can gain adaptability capabilities with minor extensions, such as the incorporation of multiple backup SSON paths in the event of unexpected failures, however adaptive learning can enable service compositions with many benefits.

Reinforcement learning can obtain the optimal composition solution by directly studying the results of earlier executions. This is performed through an online trial-and-error process that requires observing the effects of many transitions to converge to optimal composition policies. However the high dynamicity of mobile networks may render some of the reinforcement learning methods inefficient and costly. The learning process may take a prolonged period of time to converge to optimal composition solutions if the network conditions change frequently, such the case in mobile networks. We plan to investigate the feasibility of reinforcement learning in autonomic SSON compositions. The reliance in some cases on Markov Decision Processes to model service compositions can have its limitations. The learning process aims to obtain the optimal policy of the Markov decision process that delivers the best quality of service. Properties are learnt as services are executed throughout the lifecycle of a service composition. We attempt to investigate the feasibility

of such methods as MP services participate in multiple SSONs simultaneously and as non-functional properties of services dynamically change.

8.2.3. Proactive Adaptation and Seamless SSON Handovers

The high level of dynamicity associated with mobile environments requires proactive adaptations to dynamically replace single service operations or even groups of operations through QoS and movement prediction techniques. Proactive approaches of service composition attempt to predict problems in a composition before they occur. Existing approaches focus on the proactive adaptation of service compositions prior to, or during early stages of service execution. We plan to extend this research work to investigate the feasibility of applying proactive learning during service runtime without affecting current SSON sessions. This work may involve user mobility prediction and seamless handover of network connections. We provided an initial overview of a possible solution in [203]. The work integrated a framework introduced by Al Ridhawi et al. [204] [205] to provoke VHO operations to meet requirements of Service Level Agreements (SLAs) established with mobile users. The framework continuously derived optimal VHO configuration policies with respect to vertical handover initiation and network selection for SSON nodes' migration. The work discussed some of the possible solutions pertaining to predictive SSON establishment in neighboring networks prior to any vertical handover of the MC's connection. However, further research should be conducted regarding the costs, both financial and in terms of delay, associated with such approach. Moreover, we plan to address the difficult challenges of coordinating decisions between different, and possibly conflicting, autonomic managers at the node and network levels. A technique to coordinate evaluation metrics and actions among such managers should also be developed.

8.3. Research Work Limitations

As we have illustrated, the focus of the conducted research has been the development of an automatic SSON construction mechanism in mobile networks. We have adhered to the importance of meeting user quality requirements even with the unpredictability and

dynamicity of mobile environments. In this work we presented two solutions: a plan-based and a plan-free composition mechanism. Plan-based composition relied on two prerequisites: a MS-generated composition plan, and a hybrid service overlay network (H-SON). The plan restricted service discovery to descriptions clearly outlined in the MS's plan. Moreover, service discovery relied on the H-SON that linked service nodes according to their provided services and their respective QoS levels. To expand MP autonomy and free the SSON composition from preconceived plans we introduced the semantic nearness-based service discovery, comparison, and composition mechanism. However, the main limitations of the current research work can be summarized as follows:

1. The fuzzy-based QoS evaluation and comparison to MC requirements, as well as the semantic similarity/nearness evaluation of service ports introduces additional cost to SSON compositions. As presented in this dissertation, a noticeable increase in delay and computations is seen in our approach when compared to centralized compositions and even over-simplified decentralized solutions as seen in [32]. However, we clearly justified the additional costs with the benefits associated with meeting the MC service descriptive and QoS requirements while maintaining MP autonomy. Nevertheless, we plan to continue our research work to enhance the performance of our presented service discovery and compositions mechanism. Our goal is to match or exceed the performance of existing solutions while maintaining the benefits gained in our system.
2. Associating each SSON with a single MC can empower service composition with personification abilities. However simultaneous service requests from every user and the large numbers of users requiring service compositions within a network, may result in the construction of numerous SSON paths. As such, multiple copies of SSON can exist if different users require equivalent services. It is thus beneficial to share existing SSONs (if involved service nodes have the resources) than to compose new ones whenever a MC request arrives. Such a decision involves complex conditions such as MC location, MP resources, required QoS, etc...
3. Once an SSON is composed, the fact such an SSON exists in a mobile environment suggests that MP movements, degradations in QoS levels, and changes in the network environment are highly likely. Continuous monitoring and adaptation of SSONs is

needed. Do the same nodes involved in the SSON perform the required monitoring? Do we include neighboring nodes in the monitoring process? When is a path reconfiguration required? What is the cost of incorporating monitoring? Many other questions remain to be exploited and researched.

Simulations and evaluation tests performed on our presented SSON composition mechanisms show that our scheme is characterized by high levels of accuracy and stability regardless of service discovery mechanisms employed.

The empirical evaluation of the proposed system presented in this dissertation provides insight to the behavior of the proposed SSON composition solution. Results showed that our solution is characterized by high stability and could be implemented over any existing service discovery mechanisms. Moreover, the costs associated with the fuzzy QoS, semantic similarity, and semantic nearness evaluations remain within acceptable levels for the benefits gained in terms of accuracy to the MC's media flow description and QoS requirements.

List of Publications

Journal Publications

- **Y. Al Ridhawi**, A. Karmouch, “Decentralized Plan-Free Semantic-Based Service Composition in Mobile Networks,” [under revision] to IEEE Transactions on Services Computing, 2013.
- **Y. Al Ridhawi**, A. Karmouch, “QoS-based Composition of Service Specific Overlay Networks,” [under review] to IEEE Transactions in Computing, 2013.

Conference Publications

- **Y. Al Ridhawi,** I. Al Ridhawi, A. Karmouch, “An Adaptive Vertical Handover in Service Specific Overlay Networks,” in 3rd Workshop on Enablers for Ubiquitous Computing and Smart Services (EUCASS 2012), pp.418-423, 2012.
- **Y. Al Ridhawi**, I. Abdeljaouad, A. Karmouch, "Service Specific Overlay Composition and Reconfiguration," accepted by 2012 International Conference on Multimedia Computing and Systems (ICMCS), pp.479-484, 2012.
- **Y. Al Ridhawi**, K. Gajaruban, and A. Karmouch, “A Dynamic Hybrid Overlay Network for Service Compositions,” Proc. 10th Int’l Conference on Wireless Networks (ICWN’11), pp. 389-395, 2011.
- **Y. Al Ridhaw**, I. Abdeljaouad, G. Kandavanam, A. Karmouch, "An Architecture for Autonomic Management of Overlay Networks," 2nd IEEE Workshop on Multimedia Communications & Services (MCS 2011), IEEE Globecom, 2011 , pp.77-82, 2011.
- **Y. Al Ridhawi**, I. Al Ridhawi, A. Karmouch, "A Context-aware and Location Prediction Framework for Dynamic Environments," The 7th IEEE International Conference on Wireless and Mobile Computing, WIMOB 2011, pp.172-179, 2011.
- **Y. Al Ridhawi**, I. Al Ridhawi, A. Karmouch, "Policy-Based Personalized Context dissemination for Location-Aware Services," The 7th International ICST Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, pp.366-371, 2012.

- **Y. Al Ridhawi**, A. Karmouch, “Ontology-Based Negotiation Protocol and Context-Level Agreements”, in Proc. IET 4th International Conference on Intelligent Environments, pp. 1-8, 2008.
- **Y. Al Ridhawi**, H. Harroud, A. Karmouch, N. Agoulmine, “Policy Driven Context-Aware Services in Mobile Environments,” in Proc. International Conference on Innovations in Information Technology (IIT’08), pp.558-562, 2008.
- **Y. Al Ridhawi**, A. Karmouch, “Ontology-Based Context-Level Agreements and Negotiation Protocol”, Fifth International Workshop on Next Generation Networking Middleware (NGNM 2008), pp.1-6, 2008.

BIBLIOGRAPHY

[1] M. P. Papazoglou, “Service-Oriented Computing: concepts, characteristics and directions,” in Proc. 4th International Conference on Web Information Systems Engineering, pp.3-12, 2003.

[2] M. Bichier, “Service-Oriented Computing,” in Computer Journal, vol. 39, no. 3, pp.99-101, 2006.

[3] M. N. Huhns and M.P. Singh, “Service-Oriented Computing: key concepts and principles,” in IEEE Internet Computing, vol. 9, no. 1, pp.75-81, 2005.

[4] H. Petritsch. “Introduction to SOA and Web Services” in *Integrating SOA and Web Services*, River Publishers, Denmark: River Publishers, pp.9-10, 2011.

[5] G. Ding and B.K. Bhargava, “Peer-to-Peer File-Sharing Over Mobile Ad Hoc Networks,” in Proc. 2nd IEEE Annual Conference on Pervasive Computing and Communications Workshops, pp.104-108, 2004.

[6] Z. Lu, X. Gao, S. Huang, Y. Huang, “Scalable and Reliable Live Streaming Service Through Coordinating CDN and P2P,” in Proc. 17th IEEE International Conference on Parallel and Distributed Systems, pp. 581-588, 2011.

[7] C. Lee, S. Kim, S. Kang, “A Framework for Context-Aware P2P Service,” in Proc. IEEE Asia-Pacific Service Computing Conference, pp.498-502, 2011.

[8] Z. Xu, R. Min, Y. Hu, “HIERAS: a DHT based hierarchical P2P routing algorithm,” in Proc. of International Conference on Parallel Processing, pp. 187-194, 2003.

[9] M. Yang and Y. Yang, “An Efficient Hybrid Peer-to-Peer System for Distributed Data Sharing,” in IEEE Transactions on Computers, vol. 59, no. 9, pp.1158-1171, 2010.

[10] L. Subramanian, I. Stoica, H. Balakrishnan, R. Katz, “OverQoS: An Overlay Based Architecture for Enhancing Internet QoS,” in Proc. 1st Symposium on Networked Systems Design and Implementation (NSDI), pp.1-14, 2004.

[11] D. Anderson, H. Balakrishnan, F. Kaashoek and R. Morris, “Resilient Overlay Networks,” in 18th ACM Symposium on Operating Systems Principles (SoSP), pp.1-15, 2001.

[12] H. Tong, J. Cao, S. Zhang, and M. Li, “A Distributed Algorithm for Web Service Composition Based on Service Agent Model,” in IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 12, pp.2008-2021, 2011.

[13] Z. Duan, Z. Zhang, and Y.T. Hou, “Service overlay networks: SLAs, QoS, and bandwidth provisioning,” in IEEE/ACM Transactions on Networking, vol. 11, issue 6, pp.870-882, 2003.

[14] S. Schmid, S. Herborn, J. Rey, “SMART: Intelligent Multimedia Routing and Adaptation based on Service Specific Overlay Networks,” in Proceedings of Ubiquitous services and applications, EURESCOM’05, pp.69-77. 2005.

[15] I. Al-Oqily, A. Karmouch, “Automating Overlay Networks Management,” in Proc. 21st International Conference on Advanced Information Networking and Applications pp.386-393, 2007.

[16] V.X. Tran, H. Tsuji, “QoS Based Ranking for Web Services: Fuzzy Approaches,” in Proc. 4th International Conference on Next Generation Web Services Practices (NWESP’08), pp.77-82, 2008.

[17] L. Zhuang, H. YuanFei, J. WeiGuang, Z. JiangBo, G. He-Qing, “Solving Fuzzy QoS Constraint Satisfaction Technique for Web Service Selection,” in International Conference on Computational Intelligence and Security Workshops, pp.35-38, 2007.

[18] Z. HongKang, Y. XueLi, Z. GuangPing, H. Kun, “Research on Services Matching and Ranking Based on Fuzzy QoS Ontology,” in International Conference on Computational Aspects of Social Networks (CASoN), pp.579-582, 2010.

[19] R. Zhang, J. Ma, “Fuzzy QoS Management in Diff-Serv Networks,” in IEEE International Conference on Systems, Man, and Cybernetics, pp.3752-3757, 2010.

[20] D. Shin, G. J. Ahn, J. S. Park, “An Application of Directory Service Markup Language (DSML) for Role-based Access Control (RBAC),” in Proc. 26th Annual International Computer Software and Applications Conference, pp.934-939, 2002.

[21] A. Ankolekar, M. Burstein, J. R. Hobbs, O. Lassila, D. Martin, D. McDermott, S. A. Mcllraith, S. Narayanan, M. Paolucci, T. Payne, “DAML-S: Web Service Description for the Semantic Web,” in The Semantic Web – Lecture Notes in Computer Science, vol. 2342, pp.348-363, 2002.

[22] C. Zhou, L. T. Chia, B. S. Lee, “DAML-QoS Ontology for Web Services,” in IEEE International Conference on Web Services, pp.472-479, 2004.

[23] P. Rengaraju, C.H. Lung, F.R. Yu, A. Srinivasan, “On QoE Monitoring and E2E Service Assurance in 4G Wireless Networks,” in IEEE Wireless Communications, pp.89-96, 2012.

[24] P. Hershey, S. Rao, A. Narayan, “System of Systems to Provide Quality of Service Monitoring, Management and Response in Cloud Computing Environments,” in Proc. of 7th International Conference on System of Systems Engineering, pp.314-320, 2012.

[25] I. Lahyani, N. Khabou, M. Jmaiel, “QoS Monitoring and Analysis Approach for Publish/Subscribe Systems Deployed on MANET,” in Proc. 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing, pp.120-124, 2012.

[26] N. Niebert, A. Schider, H. Abramowicz, G. Malmgren, J. Sachs, U. Horn, C. Prehofer, H. Karl, “Ambient Networks: An Architecture for Communication Networks Beyond 3G,” IEEE Wireless Communications, vol. 11, no.2, pp. 14-22, 2004.

[27] P. Kersch, R. Szabo, Z. L. Kis, M. Erdei, B. Kovacs, “Self Organizing Ambient Control Space: an ambient network architecture for dynamic network interconnection,” in Proc. 1st ACM workshop on Dynamic Interconnection of Networks, pp.17-21, 2005.

[28] J. Rey, D. Lozano, S. Herborn, K. Ahmed, S. Schmid, F. Hartung, M. Kampmann, M. Kleis, “SMART: A Media Service Delivery Architecture for Ambient Networks,” in Proc. 14th IST Mobile and Wireless Communications Summit, pp.1-6, 2005.

[29] S. Chen, K. Nahrstedt, “An Overview of Quality-of-Service Routing for the Next Generation High-Speed Networks: Problems and Solutions,” in IEEE Networks, pp.64-79, 1998.

[30] Y. Cheng, W. Zhuang, “DiffServ Resource Allocation for Fast Handoff in Wireless Mobile Internet,” IEEE Communications Magazine, vol. 40, no. 5, pp.130-136, 2002.

[31] I. Mahadevan, K.M. Sivalingam, “Quality of Service Architectures for Wireless Networks: IntServ and DiffServ Models,” in Proceedings of 4th International Symposium on Parallel Architectures, Algorithms, and Networks, pp.420-425, 1999.

[32] I. Al-Oqily, A. Karmouch, “SORD: A Fault-Resilient Service Overlay for MediaPort Resource Discovery,” IEEE Transactions on Parallel and Distributed Systems, Vol.20, Issue 8, pp.1112-1125, 2009.

[33] W. Wang, D. A. Helder, S. Jamin, L. Zhang, “Overlay Optimizations for End-Host Multicast,” in Proc. Networked Group Communication, pp.154-161, 2002.

[34] B. Zhang, S. Jamin, L. Zhang, “Host Multicast: A Framework for Delivering Multicast To End Users,” in Proc. IEEE 21st Annual Joint Conference of the IEEE Computer and Communications Societies, pp.1366-1375, 2002.

[35] Y. Chawathe, S. Mccanne, and E. A. Brewer, “RMX: Reliable multicast for heterogeneous networks,” in Proc. INFOCOM’00, pp.795–804, 2000.

[36] Y. Chu, S. G. Rao, and H. Zhang, “A case for end system multicast,” in Proceedings of ACM SIGMETRICS, pp.1–12, 2000.

[37] Z. Li, P. Mohapatra, “QRON: QoS-Aware Routing in Overlay Networks,” in IEEE Journal on Selected Areas in Communications, vol.22, no.1, pp.29-40, 2004.

[38] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, H. Balakrishnan, “Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications,” in ACM SIGCOMM’01, pp.1-12, 2001.

[39] E. Kuhn, J. Riemer, L. Lechner, “XVSMP/Bayeux: A Protocol for Scalable Space Based Computing in the Web,” in Proc. 16th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise, pp. 68-73, 2007.

[40] Y. Chawathe, S. McCanne, E. A. Brewer, “RMX: Reliable Multicast for Heterogeneous Networks,” in Proc. IEEE International Conference on Computer Communications, pp.795-804, 2000.

[41] J. Jannotti, D. Gifford, K. Johnson, M. Frans Kaashoek, J. O’Toole, M. Frans, K. James, “Overcast: Reliable Multicasting with an Overlay Network,” in ACM Proceedings of 4th Conference on Symposium on Operating System Design and Implementation, vol.4, pp.197-212, 2000.

[42] N. Malouch, Z. Liu, D. Rubenstein, S. Sahu, “A Graph Theoretic Approach to Bounding Delay in Proxy-Assisted, End-System Multicast,” in Proceedings of 10th International Workshop on Quality of Service (IWQoS’02), pp.106-115, 2002.

[43] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, J. D. Kubiatwicz, “Bayeux: an architecture for scalable and fault-tolerant wide-area data dissemination,” in Proc. 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video, pp.11-20, 2001.

[44] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, “Tapestry: A Resilient Global-Scale Overlay for Service Deployment,” in IEEE Journal on Selected Areas in Communications, vol.22, no.1, pp. 41-53, 2004.

[45] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, M. Bowman, “PlanetLab: an Overlay Testbed for Broad-Coverage Services,” in ACM SIGCOMM Computer Communication Review, vol.33, no.3, pp3-12, 2003.

[46] N. Spring, L. Peterson, A. Bavier, V. Pai, “Using PlanetLab for Network Research: myths, realities, and best practices,” in ACM SIGOPS Operating Systems Review, vol.40, no.1, pp.17-24, 2006.

[47] A. Nakao, L. Peterson, A. Bavier, “A Routing Underlay for Overlay Networks,” in Proceedings of 2003 Conference on Applications, technologies, architectures, and protocols for Computer Communications (SIGCOMM’03), pp.11-18, 2003.

[48] K. Shen, “Substrate-Aware Connectivity Support for Scalable Overlay Service Construction,” in 1st USENIX/ACM Symposium on Networked Systems Design and Implementation, pp.1-14, 2004.

[49] R. Braynard, D. Kostic, A. Rodriguez, J. Chase, A. Vahdat,” Opus: an Overlay Peer Utility Service,” in Proceedings of IEEE Open Architectures and Network Programming, pp.167-178, 2002.

[50] J. Touch, “Dynamic Internet Overlay Deployment and Management Using the X-Bone,” in Computer Networks, vol.36, Issue 2-3, pp.117-135, 2001.

[51] Y. Qiao, F. Bustamante, “Structured and Unstructured Overlays Under the Microscope- A Measurement-Based View of Two P2P Systems that People Use,” in Proceedings of the USENIX Annual Technical Conference, pp.341-355 , 2006.

[52] D. Stutzbach, R. Rejaie, S. Sen, “Characterizing Unstructured Overlay Topologies in Modern P2P File-Sharing Systems,” in IEEE/ACM Transactions on Networking, vol.16, no.2, pp.267-280, 2008.

[53] M. Castro, M. Costa, A. Rowstron, “Peer-to-Peer Overlays: structured, unstructured, or both,” Microsoft Research, Cambridge, MSR-TR-2004-73, pp.1-15, 2004.

[54] X. Liu, G. Cheung, C. N. Chuah, “Structured Network Coding and Cooperative Wireless Ad-Hoc Peer-to-Peer Repair for WWAN Video Broadcast,” in IEEE Transactions on Multimedia, vol.11, no.4, pp.730-741, 2009.

[55] M. Ripeanu, I. Foster, A. Iamnitchi, “Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems,” in Proceedings of IEEE International Conference on Computer Communications (INFOCOM’03), pp.1-6, 2003.

[56] Y. Chawathe, S. Ratnasamy, L. Breslau, “Making Gnutella-like P2P Systems Scalable,” in Proceedings of ACM SIGCOMM, pp.407-418, 2003.

[57] T. Karagiannis, A. Broido, M. Faloutsos, K. Claffy, “Transport Layer Identification of P2P Traffic,” in Proceedings of Internet Measurement Conferences, pp.121-134 , 2004.

[58] K. Tutschku, “A Measurement-Based Traffic Profile of the eDonkey Filesharing Service,” in Lecture Nodes in Computer Science, vol.3015, pp.12-21, 2004.

[59] N. Leibowitz, M. Ripeanu, A. Wierzbicki, “Deconstructing the Kazaa Network,” in 3rd IEEE Workshop on Internet Applications, pp.112-120, 2003.

[60] S. Androutsellis-Theotokis, D. Spinellis, “A Survey of Peer-to-Peer Content Distribution Technologies,” in ACM Journal on Computing Surveys, vol.36, no.4, pp.335-371, 2004.

[61] A. Caniff, L. Lei, M. Ningfang, L. Cherkasova, E. Smirni, “Fastrack for Taming Burstiness and Saving Power in Multi-Tiered Systems,” in 22nd International Teletraffic Congress, pp.1-8, 2010.

[62] The Overnet File-sharing Network, available at http://www.overnet.com/, 2002

[63] P. Maymounkov, D. Mazieres, “Kademlia: A Peer-to-Peer Information System Based on the XOR Metric,” in Peer-to-Peer Systems: Lecture Nodes in Computer Science, vol.2429, pp.53-65, 2002.

[64] H. Guo, J. Liu, Z. Wang, “Frequency-Aware Indexing for Peer-to-Peer On-Demand Video Streaming,” in *IEEE International Conference on Communications*, pp. 1-5, 2010.

[65] Z. Li, G. Xie, K.W. Hwang, Z.C. Li, “Churn-Resilient Protocol for Massive Data Dissemination in P2P Networks,” in IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 8, pp.1342-1349, 2011.

[66] R. Laskowski, “SURFNET: A Program for Visualizing Molecular Surfaces, Cavities, and Intermolecular Interactions,” in Journal of Molecular Graphics, vol.13, issue.5, pp.323-330, 1999.

[67] C. Makaya, B. Falchuk, D. Chee, F.J. Lin, S. Das, M. Ito, S. Komorita, T. Chiba, H. Yokota, "Service Composition Based on Next-Generation Service Overlay Networks Architectures," in Proc. 4th IFIP International Conference on New Technologies, Mobility and Security, pp.1-6, 2011.

[68] N. Paiva Tizzo, J. Coello, E. Cardozo, "Improving Dynamic Web Service Selection in WS-BPEL Using SPARQL," in IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp.864-871, 2011.

[69] L. Baresi, S. Guinea, "Towards Dynamic Monitoring of WS-BPEL Processes," in Service-Oriented Computing: Lecture Nodes in Computer Science, vol.3826, pp.269-282, 2005.

[70] P. Wohed, W. M. P. van der Aalst, M. Dumas, A. H. ter Hofstede, "Analysis of Web Services Composition Languages: The Case of BPEL4WS," in Conceptual Modeling-Lecture Nodes in Computer Science, vol.2813, pp.200-215, 2003.

[71] C. Peltz, "Web Services Orchestration and Choreography," in IEEE Computer Society, pp.46-52, 2003.

[72] A. Barros, M. Dumas, P. Oaks, "Standards for Web Service Choreography and Orchestration: Status and Perspectives," in Business Process Management Workshops-Lecture Notes in Computer Science, vol. 3812, pp.61-74, 2006.

[73] W3C. Web Services Choreography Description Language Version 1.0 - W3C Candidate Recommendation. November 9, 2005. http://www.w3.org /TR/2005/CR-ws-cdl-10-20051109/ (accessed 2009).

[74] DAML-S: Semantic markup for web services. Published online by the DAML Services Coalition at http://www.daml.org/services/daml-s/0.9/daml-s.html, May 2003.

[75] K. Fujii, T. Suda, "Semantics-Based Dynamic Service Composition," in IEEE Journal on Selected Areas in Communications, vol.23, no.12, pp.2361-2372, 2005.

[76] J. Floch, S. Hallsteinsan, E. Stav, F. Eliassen, K. Lund, E. Gjorven, “Using Architecture Models for Runtime Adaptability,” in IEEE Software, vol.23, no.2, pp.62-70, 2006.

[77] K. Fujii, T. Suda, “Component Service Model with Semantics (CoSMoS): a New Component Model for Dynamic Service Composition,” in IEEE International Symposium on Applications and the Internet Workshops, pp.348-354, 2004.

[78] D. Ardagna, B. Pernici, “Global and Local QoS Constraints Guarantee in Web Service Selection,” in IEEE International Conference on Web Services, pp1-2, 2005.

[79] E. Park, H. Shin, “Reconfigurable Service Composition and Categorization for Power-Aware Mobile Computing,” in IEEE Transactions on Parallel and Distributed Systems, vol.19, no.11, pp.1553-1564, 2008.

[80] S. Kalasapur, M. Kumar, B.A. Shirazi, “Dynamic Service Composition in Pervasive Computing,” in IEEE Transactions on Parallel and Distributed Systems, vol.18, no.7, pp.907-918, 2007.

[81] J. Lee, H.-C. Hsiao, C.-T. King, A. Banjerjee, “Canal: a Distributed Service Composition System Using Mobile Agents,” in 3rd International Conference on Information Technology: Research and Education (ITRE’05), pp.449-453, 2005.

[82] L. Zhen, M. Parashar, “Rudder: a Rule-Based Multi-Agent Infrastructure for Supporting Autonomic Grid Applications,” in International Conference on Autonomic Computing, pp.278-279, 2004.

[83] J.R. Gallardo, D. Makrakis, H.T. Mouftah, “MARE: An Efficient Reservation-Based MAC Protocol for IEEE 802.11s Mesh Networks,” in 2nd International Conference on Advances in Mesh Networks, pp.97-102, 2009.

[84] K. Xu, M. Geria, S. Bae, “Effectiveness of RTS/CTS Handshake in IEEE 802.11 Based Ad hoc Networks,” in Ad Hoc Networks, vol.1, no.1, pp.107-123, 2003.

[85] F. Samimi, P. Mckinley, “Dynamis: Dynamic Overlay Service Composition for Distributed Stream Processing,” in Proceedings of International Conference on Software Engineering and Knowledge Engineering, pp.881-886, 2008.

[86] M. Wang, Z. Li, "sFlow: Towards Resource-Efficient and Agile Service Federation in Service Overlay Networks," in Proceedings of 24th International Conference on Distributed Computing Systems, pp.628-635, 2004.

[87] S. Kalasapur, M. Kumar, B. Shirazi, "Seamless Service Composition in Pervasive Computing Environments," in Proceedings of ACM 1st International Workshop on Multimedia Service Composition (MSC'06), pp.11-20, 2006.

[88] I. Al-Oqily, A. Karmouch, "Towards an Autonomic Management for Service Specific Overlay Networks," in IEEE Latin American Network Operations and Management Symposium, pp.1-8, 2007.

[89] I. Al-Oqily, A. Karmouch, "Policy-Based Context-Aware Overlay Networks," in 1st International Global Information Infrastructure Symposium, pp.85-92, 2007.

[90] M. Stemm, R. H. Katz, "Vertical Handoffs in Wireless Overlay Networks," in Mobile Networks and Applications Journal – Special issue: mobile networking in the Internet, vol.3, no.4, pp.335-350, 1998.

[91] S. Baolin, G. Chao, Z. Qifei, Y. Bing, L. Wei, "A Multipath On-Demand Routing with Path Selection Entropy for Ad Hoc Networks," in 9th International Conference for Young Computer Scientists, pp.558-563, 2008.

[92] S. J. Lee, M. Gerla, "Split Multipath Routing with Maximally Disjoint in Ad hoc Networks," in Proc. IEEE International Conference on Communications, pp.3201-3205, 2001.

[93] M. K. Marina, S. R. Das, "On-demand Multipath Distance Vector Routing in Ad Hoc Networks," in 9th International Conference on Network Protocols, pp.14-23, 2001.

[94] J. Kephart, D. Chess, "The Vision of Autonomic Computing," in Computer, vol.36, no.1, pp.41-50, 2003.

[95] Y. Al Ridhawi, G. Kandavanam, A. Karmouch; "A Dynamic Hybrid Service Overlay Network for Service Compositions," World Congress in Computer Science, Computer Engineering and Applied Computing, pp.1-7, 2011.

[96] Y. Al Ridhawi, I. Abdeljaouad, A. Karmouch, “Service Specific Overlay Composition and Reconfiguration,” in Proc. 3rd International Conference on Multimedia Computing and Systems (ICMCS’12), pp.479-484, 2012.

[97] Y. Al Ridhawi, I. Abdeljaouad, G. Kandavanam, A. Karmouch “An Architecture for Autonomic Management of Overlay Networks,” IEEE Globecom Workshops, pp.77-82, 2011.

[98] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, S. Weerawarana, “Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI,” in IEEE Internet Computing, vol.6, no. 2, pp.86-93, 2002.

[99] S. Herborn, Y. Lopez and A. Seneviratne, “A Distributed Scheme for Autonomous Service Composition,” Proceedings of the First ACM International workshop on Multimedia Service Composition in MSC '05, pp. 21-30, 2005.

[100] L. Kagal, T. Finin and A. Joshi, “Trust-Based Security in Pervasive Computing Environments,” in ACM Computer Journal vol.34, no.12, pp.154-157, 2001.

[101] Q. Zhang, F. Miao, Z. Yuan, Q. Zhang, Z. Fan, “Construction of a Dynamic Trust Ontology Model,” in International Conference on Computational Intelligence and Security, pp.394-398, 2008.

[102] S.K. Kim, H.J. Choi, “An Ontology Model Framework for Supply of Active Situation Decision Service,” in 4th International Conference on Computer Sciences and Convergence Information Technology, pp.1207-1212, 2009.

[103] I. Abdeljaouad, G. Kandavanam, A. Karmouch; “A Loss-Based Utility Function for Predicting IPTV Quality of Experience over an Overlay Network,” in Proc. IEEE Global Telecommunications Conference, pp.1-6, 2011.

[104] I. Abdeljaouad, A. Karmouch, “Self-Organizing Overlay Networks for Autonomic Manager Selection,” in Proc. 3rd IEEE International Workshop on Smart Communications in Network Technologies, pp.6437-6441, 2012.

[105] D. Soldani, “Means and Methods for Collecting and Analyzing QoE Measurements in Wireless Networks,” in International Symposium on World of Wireless, Mobile and Multimedia Networks, pp.529-535, 2006.

[106] D. De Vera, P. Rodriguez-Bocca, G. Rubino, “QoE Monitoring Platform for Video Delivery Networks,” in IP Operations and Management-Lecture Notes in Computer Science, vol.4786, pp.131-142, 2007.

[107] S. Balasubramaniam, J. Mineraud, P. McDonagh, P. Perry, L. Murphy, W. Donnelly, D. Botvich, “An Evaluation of Parameterized Gradient Based Routing with QoE Monitoring for Multiple IPTV Providers,” in IEEE Transactions on Broadcasting, vol.57, no.2, pp.183-194, 2011.

[108] B. Parhami, “Chordal Rings Based on Symmetric Odd-Radix Number Systems,” in Proc. International Conference on Communications in Computing, pp. 196-199, 2005.

[109] L. Narayanan, J. Opatrny, D. Sotteau, “All-to-All Optical Routing in Optimal Chordal Rings of Degree Four,” in Proc. 10th Annual ACM-SIAM Symposium on Discrete Algorithms, pp.695-703, 1999.

[110] R.F. Potthoff and S.N. Roy, “A Generalized Multivariate Analysis of Variance Model Useful Especially for Growth Curve Problems,” in Oxford Journal of Biometrika, vol.51, no.3-4, pp.313-326, 1964.

[111] A. Varga, “OMNeT++ User Manual, Version 4.1,” [online report] http://www.omnetpp.org/doc/omnetpp/manual/usman.html, 2012.

[112] I. Baumgart, B. Heep, S. Krause, “OverSim: A Flexible Overlay Network Simulation Framework,” in Proceedings of 10th IEEE Global Internet Symposium (GI '07) in conjunction with IEEE INFOCOM 2007, p.79-84, 2007.

[113] “OWL Web Ontology Language Reference,” Internet: http://www.w3.org/TR/owl-ref/ , W3C Note, 2004.

[114] G. Antoniou, F. van Harmelen, “Web Ontology Language: OWL,” in International Handbooks on Information Systems, pp.91-119, 2009.

[115] Y. Al Ridhawi, A. Karmouch, “Ontology-Based Negotiation Protocol and Context-Level Agreements”, 4th International Conference on Intelligent Environments - IE08, pp.1-8, 2008.

[116] Y. Al Ridhawi, A. Karmouch, “Ontology-Based Context-Level Agreements and Negotiation Protocol”, Fifth International Workshop on Next Generation Networking Middleware (NGNM 2008), Samos Island, Greece, pp.1-6, 2008.

[117] Y. Al Ridhawi, I. Al Ridhawi, A. Karmouch, “Policy-Based Personalized Context Dissemination for Location-Aware Services”; in Proc. 10th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, pp.366-371, 2012.

[118] Y. Al Ridhawi, H. Harroud, A. Karmouch, N. Agoulmine, “Policy Driven Context-Aware Services in Mobile Environments,” in International Conference on Innovations in Information Technology (IIT’08), pp.558-562, 2008

[119] M. Strimpakou, I. Roussaki, M. Anagnostou, “A Context Ontology for Pervasive Service Provision,” in 20th IEEE International Conference on Advanced Information Networking and Applications (AINA’06), pp.5-9, 2006.

[120] G. Tao, K. Hung, Z. Da, “A Middleware for Building Context-Aware Mobile Services,” in IEEE 59th Vehicular Technology Conference, pp.2656-2660, 2004.

[121] T. Gu, D. Zhang, H. Pung, “A Two-Tier Semantic Overlay Network for P2P Search,” in International Conference on Parallel and Distributed Systems, pp.1-8, 2007.

[122] M.A. Strimpakou, I.G. Roussaki, M.E. Anagnostou, “A Context Ontology for Pervasive Service Provision,” in 20th International Conference on Advanced Information Networking and Applications, pp.1-5, 2006.

[123] J. Wang, J. Zhang, Y. Wang, “Research on Semantic Web-Oriented Ontology Model,” in International Conference on Internet Computing in Science and Engineering, pp.258-261, 2008.

[124] S.K. Kim, H.J. Choi, “An Ontology Model Framework for Supply of Active Situation Decision Service,” in 4th International Conference on Computer Sciences and Convergence Information Technology, pp.1207-1212, 2009.

[125] X. Zhifeng, Z. YongZhao, M. Qirong, X. Liting, “Awareness Ontology Model in the Collaborative Learning Environment,” in 7th International on Computer-Aided Industrial Design and Conceptual Design, pp.1-6, 2006.

[126] X. Tao, Y. Li, N. Zhong, “A Personalized Ontology Model for Web Information Gathering,” in IEEE Transactions on Knowledge and Data Engineering, vol.23 no.4, pp.496-511, 2011.

[127] D. Roman, U. Keller, H. Lausen, J. de Brujin, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, D. Fensel, “Web Service Modeling Ontology,” in Applied Ontology Journal, vol.1, no.1, pp.77-106, 2005.

[128] M. Ford, M. Ljungberg, D.J. Van Hook, R. Shaw, E. Aubin, “Resource Brokering Service: Automatic Plan Composition and Execution,” in IEEE Military Communications Conference, pp.237-242, 2010.

[129] A. Omer, A. Schill, “Dependency Based Automatic Service Composition Using Directed Graph,” in 5th International Conference on Next Generation Web Services Practices, pp.76-81, 2009.

[130] H. Yang, Z. Li, “Modeling for Web Services Composition System with Restricted Resources,” in First International Conference on Advances in System Simulation, pp.32-37, 2009.

[131] Z. Ding, J. Wang, H. Song, “AI Planning for Web Service Automatic Composition Using Petri Nets,” in 11th International Conference on Computer Supported Cooperative Work in Design, pp.519-524, 2007.

[132] S. Seheon, K. Minkoo, “A Plan-Based Service Composition for Work Process Agent in Ubiquitous Computing,” in IEEE Asia-Pacific Services Computing Conference, pp.483-487, 2011.

[133] Y. Qing-mei, X. Peng-yan, J. Ting, H. Dong-mei, “Semantic and Heuristic Approach to the Composition of Web Services,” in International Conference on Web Information Systems and Mining, pp.175-178, 2010.

[134] B. Bonet, H. Geffner, “Planning as Heuristic Search,” in Artificial Intelligence, vol. 129 no.1-2, pp.5-33, 2011.

[135] J. Rao, X. Su, “A Survey of Automated Web Service Composition Methods,” in Semantic Web Services and Web Process Composition, vol.3387, pp.43-54, 2005.

[136] M. Klusch, A. Gerber, “Semantic Web Service Composition Planning with OWLS-XPLAN,” in Proceedings of the 1st International AAAI Fall Symposium on Agents and the Semantic Web, pp.55-62, 2005.

[137] S. Herborn, A. Seneviratne, “Service Composition for Mobile Personal Networks,” in 3rd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, pp.1-8, 2006.

[138] L. Qiu, Z. Shi, F. Lin, “Context Optimization of AI Planning for Services Compositions,” in IEEE International Conference on e-Business Engineering, pp.610-617, 2006.

[139] K. Ren, X. Liu, J. Chen, N. Xiao, J. Song, W. Zhang, “A QSQL-based Efficient Planning Algorithm for Fully-automated Service Composition in Dynamic Service Environments,” in IEEE International Conference on Services Computing, pp.301-308, 2008.

[140] M. Klusch, A. Gerber, “Fast Composition Planning of OWL-S Services and Application,” in Proceedings of the European Conference on Web Services, pp.181-190, 2006.

[141] L.A. Zadeh, “Fuzzy Sets,” in Information and Control, vol.8 no.3, pp.338-353, 1965.

[142] “Fuzzy Logic Fundamentals: Chapter 3,” [online report] Available at: http://www.ece.msstate.edu/courses/ece4723/fall11/notes/FL_chapter.pdf, 2001.

[143] P. Bosc, E. Damiani, M. Fugini, “Fuzzy Service Selection in a Distributed Object-Oriented Environment,” in IEEE Transactions on Fuzzy Systems, vol.9, no.5, pp.682-698, 2001.

[144] F. Fenza, V. Loia, S. Senatore, “Improving Fuzzy Service Matchmaking through Concept Matching Discovery,” in IEEE International Fuzzy Systems Conference, pp.1-6, 2007.

[145] R. Cheung, G. Yao, J. Cao, A. Chan, “A Fuzzy Service Adaptation Engine for Context-Aware Mobile Computing Middleware,” in International Journal of Pervasive Computing and Communications, vol.4, no.2, pp.147-165, 2008.

[146] C. J. Chien, H. H. Tsai, “Using Fuzzy Numbers to Evaluate Perceived Service Quality,” in Fuzzy Sets and Systems, vol.116, no.2, pp.289-300, 2000.

[147] P. Xiong, Y. Fan, “QoS-Aware Web Service Selection by a Synthetic Weight,” in 4th International Conference on Fuzzy Systems and Knowledge Discovery, pp.632-637, 2007.

[148] P. Cingolani and J. Alcala-Fdez, “jFuzzyLogic: A Robust and Flexible Fuzzy-Logic Inference System Language Implementation,” in IEEE World Congress on Computational Intelligence, pp.1090-1097, 2012.

[149] E. H. Mamdani, “Twenty Years of Fuzzy Control: Experiences Gained and Lessons Learnt,” in 2nd IEEE International Conference on Fuzzy Systems, vol.1, pp.339-344, 1993.

[150] I. Muller, R. Kowalczyk, P. Braun, “Towards Agent-Based Coalition Formation for Service Composition,” in Proc. International Conference on Intelligent Agent Technology, pp.73-80, 2006.

[151] X. Shao, L. H. Ngoh, T. K. Lee, T. Y. Chai, L. Zhou, J. C. Teo, “Multipath Cross-Layer Service Discovery (MCSD) for Mobile Ad Hoc Networks,” in IEEE Asia-Pacific Service Computing Conference, pp. 408-413, 2009.

[152] H. W. Tsai, T. S. Chen, C. P. Chu, “Service Discovery in Mobile Ad Hoc Networks Based on Grid,” in IEEE Transactions on Vehicular Technology, vol. 58, no. 3, pp.1528-1545, 2009.

[153] W. Bethea, R. Cole, P. Harshavardhana, “Automated Discovery of Information Services in Heterogeneous Distributed Networks,” in IEEE Military Communications Conference, pp.1-6, 2008.

[154] X. Wang, C. Liu, Z. Yang, “An Efficient Semantic Web Service Discovery Algorithm in DHT-based P2P Network,” in 1st International Conference on Future Information Networks, pp.188-193, 2009.

[155] M. Cai, A. Chervenak, M. Frank, “A Peer-to-Peer Replica Location Service Based on a Distributed Hash Table,” in Proc. ACM/IEEE Conference on Supercomputing, pp.56-68, 2004.

[156] D. Tam, R. Azimi, H. A. Jacobsen, “Building Content-Based Publish/Subscribe Systems with Distributed Hash Tables,” in Databases, Information Systems, and Peer-to-Peer Computing, vol.2944, pp.138-152, 2004.

[157] Q. Lin, R. Rao, M. Li, “DWSDM: A Web Services Discovery Mechanism Based on a Distributed Hash Table,” in 5th International Conference on Grid and Cooperative Computing Workshops, pp.176-180, 2006.

[158] C. Schmidt, M. Parashar, “A Peer-to-Peer Approach to Web Service Discovery,” in Journal on World Wide Web, vol.7. no.2, pp.211-229, 2004.

[159] S. McIlraith and D. Martin, “Bringing Semantics to Web Services,” IEEE Intelligent Systems, vol. 18, no. 1, pp.90-93, 2003.

[160] Y. Yamato and H. Sunaga, “Context-Aware Service Composition and Component Change-over using Semantic Web Technologies,” in IEEE International Conference on Web Services, pp.687-694, 2007.

[161] Z. Zhang, W. Li, Z. Wu, W. Tan, “Towards an Automata-based Semantic Web Services Composition Method in Context-aware Environment,” in IEEE 9th International Conference on Services Computing, pp.320-327, 2012.

[162] X. Zhang, H. Miao, H. Zeng, “The Syntactic and Semantic Model of Web Services Composition Based Category,” International Conference on Advanced Computer Theory and Engineering, pp.444-449, 2008.

[163] I. Elgedawy, Z. Tari, M. Winikoff, “Exact Functional Context Matching for Web Services,” in Proc. 2nd International Conference on Service Oriented Computing, pp.143-152, 2004.

[164] R. Akkiraju, B. Srivastava, A. A. Ivan, R. Goodwin, T. Syeda-Mahmood, “SEMAPLAN: Combining Planning with Semantic Matching to Achieve Web Service Composition,” IEEE International Conference on Web Services, pp.37-44, 2006.

[165] M. Paolucci, T. Kawamura, T. R. Payne, K. Sycara, “Semantic Matching of Web Services Capabilities,” in The Semantic Web, vol.2342, pp.333-347, 2002.

[166] B. Li, K. Nahrstedt, “A Control-Based Middleware Framework for Quality-of-Service Adaptations,” in IEEE Journal on Selected Areas in Communications, vol.17, no.9, pp.1632-1650, 1999.

[167] G. Ting, W. Haiyang, Z. Naihui, L. Fei, “An Improved Way to Facilitate Composition-Oriented Semantic Service Discovery,” International Conference on Computer Engineering and Technology, pp.156-160, 2009.

[168] K. Iqbal, M. L. Sbodio, V. Peristeras, G. Giuliani, “Semantic Service Discovery using SAWSDL and SPARQL,” 4th International Conference on Semantics, Knowledge and Grid, pp.205-212, 2008.

[169] E. Sirin, B. Parsia, J. Hendler, “Filtering and Selecting Semantic Web Services with Interactive Composition Techniques,” in IEEE Intelligent Systems, vol. 19, no. 4, pp.42-49, 2004.

[170] B. Adryan, R. Schuh, “Gene-Ontology-based Clustering of Gene Expression Data,” in Oxford Journal of Life Sciences and Mathematics and Physical Sciences, vol.20, no.16, pp.2851-2852, 2004.

[171] J. Liu, V. Issarny, “Enhanced Reputation Mechanism for Mobile Ad Hoc Networks,” in Journal of Trust Management, vol.2995, pp.48-62, 2004.

[172] D. L. Rubin, “Creating and Curating a Terminology for Radiology: Ontology Modeling and Analysis,” in Journal of Digital Imaging, vol.21, no.4, pp.355-362, 2008.

[173] A.V. Paliwal, B. Shafiq, J. Vaidya, H. Xiong, N. Adam, “Semantics-Based Automated Service Discovery,” IEEE Transactions on Services Computing, vol. 5, no. 2, pp.260-275, 2012.

[174] M. Wang, X. Li, X. Qiao, “Semantic Web Service Discovery Based on User Preference Cluster,” in 3rd IEEE International Conference on Broadband Network and Multimedia Technology, pp.939-944, 2010.

[175] T. Kawamura, J. A. De Blasio, T. Hasegawa, M. Paolucci, K. Sycara, “Public Deployment of Semantic Service Matchmaker with UDDI Business Registry,” in Journal on the Semantic Web, vol.3298, pp.752-766, 2004.

[176] S. Ben Mokhtar, D. Preuveneers, N. Georgantas, V. Issarny, Y. Berbers, “EASY: Efficient Semantic Service Discovery in Pervasive Computing Environments with QoS and Context Support,” in Journal of Systems and Software, vol.81, no.5, pp.785-808, 2008.

[177] A. Haller, E. Cimpian, A. Mocan, E. Oren, “WSMX – A Semantic Service-Oriented Architecture,” in Proc. IEEE International Conference on Web Services, vol.1, pp.321-328, 2005.

[178] M. E. Falou, M. Bouzid, A.I. Mouaddib, T. Vidal, “Automated Web Services Composition: A Decentralized Multi-Agent Approach,” IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, pp.387-394, 2009.

[179] J. Wang, “Exploiting Mobility Prediction for Dependable Service Composition in Wireless Mobile Ad Hoc Networks,” IEEE Transactions on Services Computing, vol. 4, no. 1, pp.44-55, 2011.

[180] M. Pistore, F. Barbon, P. Bertoli, D. Shaparau, P. Traverso, “Planning and Monitoring Web Service Composition,” in Artificial Intelligence: Methodology, Systems, and Applications, vol.3192, pp.106-115. 2004.

[181] E. Sirin, B. Parsia, D. Wu, J. Hendler, D. Nau, “HTN Planning for Web Service Composition using SHOP2,” in Web Semantics: Science, Services and Agents on the World Wide Web, vol.1, no.4, pp.377-396, 2003.

[182] F. Wagner, F. Ishikawa, S. Honiden, “QoS-aware Automatic Service Composition by Applying Functional Clustering,” IEEE International Conference on Web Services, pp.89-96, 2011.

[183] C. Platzer, F. Rosenberg, S. Dustdar, “Web Service Clustering Using Multidimensional Angles as Proximity Measures,” in ACM Transactions on Internet Technology, vol.9, no.11, pp.11.1-11.26, 2009.

[184] W. Liu, W. Wong, “Discovering Homogenous Service Communities through Web Service Clustering,” in Service-Oriented Computing: Agents, Semantics, and Engineering, vol.5006, pp.69-82, 2008.

[185] R. Nayak, B. Lee, “Web Service Discovery with Additional Semantics and Clustering,” in IEEE/WIC/ACM International Conference on Web Intelligence, pp.555-558, 2007.

[186] V. R. Chifu, C. B. Pop, I. Salomie, M. Dinsoreanu, V. Acretoaie, T. David, “An Ant-inspired Approach for Semantic Web Service Clustering,” in 9th IEEE Roedunet International Conference, pp.145-150, 2010.

[187] G. Veneziano, “Pre-bangian Origin of Our Entropy and Time Arrow,” in Physics Letters B, vol.454, no.1-2, pp.22-26, 1999.

[188] M. A. Rodriguez and M. J. Egenhofer, “Determining Semantic Similarity Among Entity Classes from Different Ontologies,” IEEE Transactions on Knowledge Data Engineering, vol. 15, no. 2, pp.442-456, 2003.

[189] H. Al-Mubaid, H. A. Nguyen, “Measuring Semantic Similarity Between Biomedical Concepts Within Multiple Ontologies,” IEEE Transactions on Systems, Man, and Cybernetics, vol. 39, no. 4, pp.389-398, 2009.

[190] M. Gan, X. Dou, D. Wang, R. Jiang, “DOPCA: A New Method for Calculating Ontology-based Semantic Similarity,” 10th IEEE/ACIS International Conference on Computer and Information Science, pp.110-115, 2011.

[191] W. N. Lee, W. Bridewell, A. K. Das, “Comparison of Semantic Similarity Measures for Application Specific Ontology Pruning,” 1st IEEE International Conference on Healthcare Informatics, Imaging and Systems Biology, pp.97-103, 2011.

[192] P. Resnik, “Using Information Content to Evaluate Semantic Similarity Taxonomy,” Proceedings of the 14th International Joint Conference on Artificial Intelligence, pp.448-453, 1995.

[193] Y. Guisheng, S. Qiuyan, “Research on Ontology-Based Measuring Semantic Similarity,” International Conference on Internet Computing in Science and

Engineering, pp.250-253, 2008.

[194] S. Xia, Z. Hu, Q. Niu, “An Approach of Semantic Similarity Measure Between Ontology Concepts Based on Multi Expression Programming,” 6th Web Information Systems and Applications Conference, pp.184-188, 2009.

[195] Z. Wu, M. Palmer, “Verb Semantics and Lexical Selection,” Proceedings of Associated Computing Linguistics, pp.133-138, 1994.

[196] K. Anyanwu, A. Maduko, A. Sheth, “SemRank: Ranking Complex Relationship Search Results on the Semantic Web,” in Proc. 14th International Conference on World Wide Web, pp.117-127, 2005.

[197] T. Adamusiak, T. Burdett, N. Kurbatova, K. Velde, N. Abeygunawardena, D. Antonakaki, M. Kapushesky, H. Parkinson, M. Swertz, “OntoCAT – Simple Ontology Search and Integration in Java, R and REST/JavaScript,” BMC Bioinformatics, pp.1471-2105, 2011.

[198] A. Medina, A. Lakhina, I. Matta, J. Byers, “Brite: Universal Topology Generation from a User’s Perspective,” Tech. rep. UMI Order Number: 2001-003, Boston University, pp.1-47, 2001.

[199] F. Agboma, A. Liotta, “QoE-Aware QoS Management,” in Proc. 6th International Conference on Advances in Mobile Computing and Multimedia, pp.111-116, 2008.

[200] A. S. Patrick, J. Singer, B. Corrie, S. Noel, K. El Khatib, B. Emond, T. Zimmerman, S. Marsh, “A QoE Sensitive Architecture for Advanced Collaborative Environments,” in 1st International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks, pp.319-322, 2004.

[201] M. Siller, J. C. Woods, “QoS Arbitration for Improving the QoE in Multimedia Transmission,” in International Visual Information Engineering, pp.238-241, 2003.

[202] S. Chaari, Y. Badr, F. Biennier, “Enhancing Web Service Selection by QoS-based Ontology and WS-Policy,” in Proc. ACM Symposium on Applied Computing, pp.2426-2431, 2008.

[203] Y. Al Ridhawi, I. Al Ridhawi, A. Karmouch, “An Adaptive Vertical Handover in Service Specific Overlay Networks,” in 3rd Workshop on Enablers for Ubiquitous Computing and Smart Services (EUCASS 2012), pp.418-423, 2012.

[204] I. Al Ridhawi, N. Samaan, A. Karmouch, “Simulator Assisted Joint Service-Level Agreement and Vertical Handover Adaptation for Profit Maximization,” in IEEE/IPSJ 12th International Symposium on Applications and the Internet (SAINT), pp.74-82, 2012.

[205] I. Al Ridhawi, N. Samaan, A. Karmouch, “A Policy-Based Simulator for Assisted Adaptive Vertical Handover,” in IEEE International Symposium on Policies for Distributed Systems and Networks, pp.41-48, 2011.