DYNAMIC COMPOSITION AND MANAGEMENT OF VIRTUAL DEVICES FOR AD HOC MULTIMEDIA SERVICE DELIVERY

By

Eric Karmouch

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the Doctor of Philosophy degree in
Computer Science

School of Information Technology and Engineering (SITE)
Faculty of Engineering
University of Ottawa

Abstract

Pervasive computing implies the invisibility of the technology involved in providing ubiquity, such that technology is integrated into the environment and non-intrusive. In such a manner, computing and networking resources become diffused into physical environments, enabling users to exploit their provided functionalities such that functionality is distributed, enabling it to be controlled, monitored, managed, and extended beyond what it was initially designed to do. Moreover, computer awareness moves towards user-centricity, whereby systems seamlessly adapt to the characteristics, preferences, and current situations of users and their respective surrounding environments. Users exploit such functionalities in the form of a virtual device, whereby a collection of heterogeneous devices in the vicinity of the user are behaving as one single homogeneous device for the benefit of the user in solving some given task.

This dissertation investigates the problem of dynamic composition and management of virtual devices for ad hoc multimedia service delivery and proposes an autonomous policy driven framework for virtual device management. The framework consists of a hierarchical structure of distributed elements, including autonomic elements, all working towards the self-management of virtual devices. The research presented in this dissertation addresses the functionalities of these components.

More specifically, contributions are made towards the autonomous management of virtual devices, moving away from infrastructure based schemes with heavy user involvement to decentralized and zero touch (i.e., no user involvement) solutions. In doing so, the components and methodology behind a policy-driven autonomous framework for the dynamic discovery, selection, and composition of multimodal multi-device services are presented. The framework operates in an ad hoc network setting and introduces a Service Overlay Network (SON) based definition of a virtual device.

Furthermore, device and service discovery, composition, integration, and adaptation schemes are designed for Mobile Ad hoc Network Environments (MANETs) enabling users to generate, on-the-fly, complex strong specific systems, embedding in a distributed manner, QoS models providing compositions that form the best possible virtual device at the time of need.

Experimental studies are presented to demonstrate the performance of the proposed schemes.

Dedication

To my wife, Kulsoom, and my son, Aydan.

Acknowledgements

I would like to offer my profound thanks to my supervisor, Prof. Amiya Nayak, for his kind supervision, constructive discussions and continuous motivation. He is not only a thesis advisor, but also a role model to me; without his continued support, encouragement, and especially his guidance and mentoring, it would not have been possible to complete this work. Moreover, I would like to thank him for giving me the freedom to strive for innovation.

I would like to thank my colleagues for their fruitful discussions, advice, and feedback during my time at the University of Ottawa. I would also like to thank my doctoral committee for their time, effort, and comments.

I would like to thank the Natural Sciences and Engineering Research Council of Canada for believing in my research and giving me the financial support to pursue it.

I would like to thank my parents for their encouragement and support, and my wife, Kulsoom; her patience and encouragement is a source of continuous motivation.

Contents

List of Figures

List of Tables

Abbreviations

3GPP	3rd Generation Partnership Project
3PCC	3rd Party Call Control
AA	Analyzing Agent
AC	Autonomic Computing
A-CM	Autonomic Composition Manager
ANI	Ambient Network Interface
A-NM	Autonomic Node Manager
ASI	Ambient Service Interface
Auto-VDMA	Autonomous Virtual Device Management Architecture
A-VDM	Autonomic Virtual Device Manager
DPEL4WS	Business Process Execution Language for Web Services
CDN	Content Distribution Network
COA	Care of Address
CRA	Conflict Resolution Agent
CRD	Capability Reconciliation DistCSP
DAML-S	DARPA Agent Markup Language for Services
DARPA	Defense Advanced Research Projects Agency
DistCSP	Distributed Constraint Satisfaction Problem
DMTF	Distributed Management Task Force
DSC	Distributed Service Composition
DVDC	Distributed Virtual Device Composition
FA	Foreign Agent

GPRS	General Packet Radio Service
HA	Home Agent
ICL	Intelligent Control Loop
IETF	Internet Engineering Task Force
iHCI	Implicit Human Computer Interaction
IM	Interface Manager
IMS	IP Multimedia Subsystem
IMT	International Mobile Telecommunications
IP	Internet Protocol
ISP	Internet Service Provider
ITU	International Telecommunication Union
LAN	Local Area Network
MANET	Mobile Ad Hoc Network Environment
MC	Media Client
MCF	Monetary Cost Function
MIP	Mobile IP
MN	Mobile Node
MP	Media Port
MS	Media Server
MSC	Media Service Client
MSP	Media Service Port
MSS	Media Service Server
NCF	Network Cost Function
OWL	Web Ontology Language

PAN	Personal Area Network
PEA	Policy Enforcement Agent
PGA	Policy Generator Agent
QoS	Quality of Service
RIT	Resource Interface Touchpoint
RS	Requesting Source
RTP	Real-time Transport Protocol
SIP	Session Initiation Protocol
SMART	Smart Media Routing and Transport
SOA	Service Oriented Architecture
SON	Service Overlay Network
SONM	Service Overlay Network Manager
SP	Service Provider
SSON	Service Specific Overlay Network
TCP	Transmission Control Protocol
TCSpec	Task Composition Specification
UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol
UML	Unified Modeling Language
UPnP	Universal Plug and Play
URI	Uniform Resource Identifier
VD	Virtual Device
VD-DistCSP	Virtual Device Distributed Constraint Satisfaction Problem

VoIP	Voice Over IP
WAN	Wide Area Network
WSDL	Web Service Definition Language
WSFL	Web Services Flow Language
XML	Extensible Markup Language

Chapter 1

Introduction

Since the advent of pervasive computing [1], the boundaries of distributed computing have changed significantly, particularly with the appearance of mobile computing and wireless networks. A new generation of handheld devices has evolved, revolutionizing how one can access and disseminate information (e.g., smart phones, set-top-boxes, netbooks, portable media players, and gaming consoles), along with network paradigms that push the concept of computing without the presence of stable network backbones. Furthermore, advancements in microcontrollers have made devices cheaper, better performing, power efficient, highly reliable, and increasingly integrated with network interfaces of various types. The realization of continuously available network connectivity coupled with an increase of online data availability is causing a consumer demand for the seamless integration of networked services into their lifestyle. This is driving network and computing technology into everyday objects (e.g., refrigerators, speakers, televisions, washing machines, and thermostats) and bringing to fruition the concept of the *networked appliance* [2].

A networked appliance is defined as a dedicated function consumer device with an embedded processor, a network connection, and the ability to disperse its capabilities within the network, allowing other devices to combine and use them as part of a *virtual device*, i.e., a collection of heterogeneous devices in the vicinity of the user that are behaving as one single homogeneous device for the benefit of the user in solving some given task managed with little to no human involvement. A networked appliance's

functionality is distributed, enabling it to be controlled, monitored, managed, and extended beyond what it was initially designed to do. The virtual device concept raises important problems such as discovering appropriate devices; deciding what devices to compose; composing discovered heterogeneous devices into a homogeneous service; and, managing a formed virtual device once composed. Although solutions do exist in terms of network resource discovery, service provision, and device interoperability, they remain typically oversimplified, over-constrained, user involvement heavy, and/or are not designed to consider the dynamic device management aspect of the virtual device concept. Furthermore, such solutions are generally designed for infrastructure-based pervasive environments.

Pervasive environments can be classified into two types: *infrastructure-based* and *infrastructure-less* or *ad hoc*. An environment having access to computing elements in the wired infrastructure through gateways, proxies, and base stations is considered infrastructure-based, as opposed to an infrastructure-less environment, whereby network connections are created and broken with peers as needed on-the-fly. To date, the majority of research taking place in pervasive computing addresses infrastructure-based environments with very little work being done in the domain of infrastructure-less pervasive environments. It is clear that infrastructure-based solutions offer advantageous solutions in environments where users spend a majority of their time, such as the home or office; however, with pervasive computing taking an anytime anywhere approach, attention should also be given to environments a user encounters throughout the day where infrastructure-based solutions are impractical. This includes environments where the user would not spend very much time or frequent very often, like an airport lounge, a commuter vehicle, a store front, or a café. The idea is for such environments to dynamically and quickly facilitate the short lived sharing and usage of resources, data, and services on peer devices to enable maximum benefit of the computation rich vicinity. Existing centralized or semi-centralized solutions offer solutions in infrastructure-based environments, but are unsuitable for infrastructure-less ad-hoc environments, due to there dependence on stable network backbones.

The work presented in this thesis aims at developing an autonomous management framework for the discovery, composition, and adaptation of virtual devices within

mobile ad hoc network environments. This chapter briefly discusses different aspects of virtual device management and focuses on challenges of networked media, device and service management, and service discovery and composition. It further identifies the motivations behind the proposed work. Subsequently, the proposed management architecture is briefly described and thesis contributions are outlined. Finally, the organization of the remainder of the thesis is presented.

1.1 Virtual Device Challenges

In this section, we outline some of the challenges posed in the virtual device concept, which include those arising from networked media, device and service management, and service discovery and composition.

On a daily basis end users are now confronted with a wide variety of media services and applications providing access to *networked media* (e.g., text, image, audio, and video) [3], which is produced, distributed, shared, managed, and consumed utilizing the Internet through various access technologies (e.g., WiFi, WiMAX, GPRS, etc.). As a result, completely changed is the way people work, live, and play. Moreover, networked media is evolving how QoS is defined, measured, and evaluated, with a heavy demand on a user's perceptual experience requiring both objective and subjective methods of measurement and evaluation [4]. With the pace of innovation growing rapidly, new models of interaction and cooperation able to support enhanced levels of quality of experience and novel on-the-move type applications including collaborative environments, personalized services, and online gaming are bringing a number of challenges to media delivery. Such challenges include seamless multi-access connectivity enabling unfettered access to media services regardless of location; service and content providers delivering services to a broad variety of end-devices; and, intelligent media routing.

In addition, the value of super-appliances (i.e., traditional static and portable personal computers), or single complex appliances providing the functionality of many simpler special purpose devices, is quickly diminishing for the implementation of specific purpose tools scattered throughout user environments working together to

provide enhanced user experience and quality of service (e.g., lightweight portable media players and/or communication devices, netbooks, set-top-boxes, game consoles, embedded flat-panel displays, touch screens and etc.) [5]. The objective is for users to complete increasingly complex tasks more quickly, accurately, and with greater satisfaction. Such an objective requires the efforts of multiple strong specific systems forming a suite of strong specific tools, which is feasible as long as the domain of the tools remain relatively simple. However, if the domain is broad and complex such as multimedia consumption across heterogeneous devices in the user's vicinity, then the collective complexities of specialized tools can quickly overwhelm the user. An issue, which is resolved through interaction management allowing for a context-dependent interaction, based on system models including discourse, user, and task modeling in removing this burden from the user through the design of digital, strong specific, networked tools.

From a practical perspective this concept involves the convergence of a range of computing devices with network capabilities and software, seamlessly adapting to surrounding environments. Such devices may vary from resource rich devices such as a computer workstation or laptop, to increasingly constrained devices such as smart phones or mobile media devices. Constraints could appear in the form of processing power, capacity, load, display capabilities, battery power, and bandwidth, for example. Moreover, devices may be stationary (e.g., large screen displays or surround sound unit) or mobile (e.g., personal media player). However, all such devices shall provide hardware and/or software functionalities to the environment through the connection of heterogeneous networks (e.g., WAN, LAN, and PAN). A user carrying a pervasive device moves from environment to environment building pervasive applications realizing user tasks in the form of virtual devices combining the functionalities of pervasive devices and adapting the resulting combination to the specifics of each environment. The main challenges of such a vision can be summarized as follows:

- *Environment heterogeneity*: The fact that pervasive devices and their functionalities are heterogeneous in terms of underlying technologies restricts their integration in realizing user tasks.

- *Environment dynamics*: As new devices may appear in the network and other devices disappear due to lack of resources or transmission range, for example, the environment is quite dynamic, perceived in terms of the number and lifetime of pervasive functionalities a user can access at a specific time and location.
- *Device resource constraints*: With high amounts of mobility, the probability of the realization of user tasks involving devices with limited resources is high and must be considered.
- *User centrism*: Ensuring that selected functionalities best conform to the user's needs is of highest priority, as in pervasive environments the user is the center of attention. It is imperative that the user be served as seamlessly and as naturally as possible, with efficient solutions involving minimal user intervention and acceptable response times.

1.2 Motivation

In motivating the thesis research, we visit the various limitations in research work with respect to the Service Oriented Architecture paradigm, service discovery and composition, and a new and innovative routing and transport architecture.

Pervasive environments can be modeled using the Service-Oriented Architecture (SOA) [6]. The SOA abstracts pervasive functionalities as services, whereby services are independent entities with well defined interfaces able to be accessed without any knowledge of their underlying technologies. Out of SOA have emerged various works [7-11] touching on the various aspects of the pervasive computing vision including mobility, ubiquity, heterogeneity, ad hoc nature, user centrism, and dynamics. In terms of *mobility*, introduced is the notion of application mobility, whereby applications are associated to users, and as users move between environments, composed applications are saved, suspended, and resumed. However, current methods are not elaborated and rely on centralized and commonly accessible file systems. In terms of *ubiquity*, various works deploy rich functionality into multiple centralized environments interconnected through file sharing or rely on the universality of the Internet to provide ubiquitous environments. These two techniques limit the range of deployment or trade range of

deployment for less rich environments, respectively. The issue of *heterogeneity* is often not handled as most works impose homogeneous system infrastructures, although some works restrict themselves to standardized web services, making solutions slightly more pervasive. In terms of achieving *ad hoc nature*, most works tend to employ service discovery and application mapping. However, heterogeneity issues exist with regards to the solutions being restricted to compositions limited to functionally and syntactically compatible service components. *User centrism* is present in most works in terms of a system's reactive or proactive behavior with regards to interpreting user preferences, context, and intent, in order to configure computing environments to support user tasks. However, proactively exploiting a user's context and intent remains a challenge. Moreover from a virtual device perspective, existing works rely far too heavily on the development of enabling platforms, and for the most part are completely centralized. It is important to move away from infrastructure based solutions to the problem and deal strictly with mobile ad hoc network environments.

In terms of service discovery and composition architectures, the majority of solutions [12-17] are designed for wired infrastructures, utilizing concepts such as central lookup servers for service registration, discovery, and composition, assuming stable nodes connected by reliable communication channels. Such techniques often involve preconfigured composition managers residing on high resource (e.g., memory, bandwidth, processing power) dedicated machines performing service coordination and management. Moreover, these approaches do not cater to highly pervasive, mobile, and ad-hoc environments, leaving themselves vulnerable to issues such as central points of failure, mobility, and fault management.

The above issues being the case, new and innovative media routing and transport architectures are emerging in the area. One such architecture is embedded in the Ambient Networks Integrated Project [18] and is called the Smart Media Routing and Transport architecture [19] or SMART for short. The root of the architecture is a common control layer for various network types, providing flexible and dynamic network configuration, able to supply end users with seamless multi-access connectivity ensuring the best possible network connection at all times. Amalgamated into this vision are broad amounts of heterogeneity in terms of access networks, terminals, network

interfaces, users, signaling and transport protocols, applications, and services making media routing and transport a significant aspect of the work. With a large spectrum of heterogeneity incorporated into the scope of the work, necessary is the need for multimedia data needing to be proactively cached, trans-coded, split, synchronized, translated, filtered, or transformed in some way or another before being delivered. A main theme of the work is the need to cater and adapt media to the user and their environment, with an emphasis on mobility. SMART points out that media shall always be routed to the most suitable end devices, and that media will be adapted to the capabilities of respective end devices. However, how such devices and respective capabilities are discovered and composed is not addressed. Nor is the need for the intelligent and automatic composition and management of devices and the services they provided (e.g., audio/video output services) to create the best possible service configurations at the time of need, with limited to zero human intervention.

In this thesis we attempt to address some of the issues brought to the forefront in this Section. In the following Section, we outline the thesis research.

1.3 Thesis Objectives

The thesis research work approaches the issue of virtual device management as a four stage process tailored to ad hoc networks, whereby every stage is infused with QoS functionality enabling the characteristics, preferences, and current situations of users and their respective surrounding environments to remain a top priority in all aspects of management. In doing so, a novel service overlay network (SON) based definition of a virtual device emerges and is applied to an autonomous policy driven framework for virtual device management.

The first stage of the process is the broker arbitration phase. With the dynamic nature and purpose of ad hoc networks, connected nodes (i.e., devices) exhibit a great deal of heterogeneity in terms of battery power, computational power, and computational load. Spreading the work of virtual device management into the network not only benefits resource poor devices, but increases the probability of discovering and exploiting services in a more efficient manner. The broker arbitration phase provides a method of

electing a node in the network best suited to handle the specific taxing needs of a user in terms of discovering and composing the best possible virtual device at the time of need. Not only is the physical resource capacity of the broker considered, but so is its ability to provide the highest degree of match with regards to the user's desired task.

The second stage of the process is the service discovery phase. At any time within the network, a plethora of composition permutations may exist satisfying the basic needs of a user's task. Simply discovering a permutation that "fits," does not necessarily satisfy the needs of the user. Best efforts must be taken to identify all possible composition permutations and determine which of these best meets the user's needs. Moreover, this procedure should be iterative, always striving to enhance its decision. The service discovery phase provides such an attempt and is approached and studied using various techniques.

The third stage of the process is the service integration phase. At the point where this phase is executed, the respective devices and services for composition satisfying the needs of the user have been found. However, it is not a safe assumption that these services can be seamlessly composed, reason being that the presence of capability differences between devices (e.g., supported codes, resolutions, and bandwidth requirements) is relatively high in an ad hoc network. Moreover, in order for a virtual device to be composed, the splitting or joining of media may be required. The service composition phase deals with this issue by building and extending from the discovered services and devices a service overlay network containing the necessary value-added processing, such as media transformation, splitting, joining, and routing.

The final state of the process is the adaptation phase. The dynamic nature of the environment whereby virtual devices are envisioned requires a composed virtual device to deal with potential post-composition adaptation. For example the virtual device may need to adjust to changing network attributes such as throughput, delay, and jitter.

The incorporation of the above four phases is carried out through a policy-driven framework composed of a hierarchical structure of distributed elements, including autonomic elements, all working towards the self-management of virtual devices. The objectives of the thesis research can be summarized as follows.

- *To be aware and perceptual of the user's continuously changing environment*. This is of particular importance in a mobile setting, such that the user's environment is continuously changing. In order to provide the virtual device concept, the system must constantly be aware of what devices are around the user for potential service delivery.
- *To instantly recognize the expectation of the user*. The system must adapt to the individual. In a virtual device approach, the system must calculate the user's expectations based on various dynamic and static contexts such as proximity (e.g., location, available devices), user profiles (e.g., preferences), observations (e.g., temporal user habits), and etc.
- *To instantiate a seamless composed service with respect to the user's given circumstance*. Once information relating to the user's environment and expectations has been collected, it must be associated with a service instance that meets the user's circumstance.
- *To provide the user with the best possible virtual device at the time of need*. Building on the previous objective, the instantiated service must not be any random fitting service instance, but a calculated choice ensuring the best possible service instance that can be provided given the user's current circumstance and expectations.
- *To provide autonomous management*. The above objects must be satisfied in a reactive manner, whereby self-configuration and self-adaptability are crucial.
- *To simplify human management tasks*. It is imperative for the creation, maintenance, and termination of a virtual device from a user perspective to be "zero touch," such that user involvement is limited or none.

1.4 Summary of Contributions

The major thesis contributions can be summarized as follows.

1. A novel hierarchical framework for the autonomous management of virtual devices in mobile ad hoc networks. The framework identifies the necessary components in

order to automate the management tasks and provides a service overlay network based definition of a virtual device.

2. A distributed broker-based service composition protocol tailored for virtual device composition in MANETs. The protocol contributes a broker arbitration scheme electing a broker in a region providing the greatest probability for discovering services providing the highest degree of match with regards to a user's desired task. Integrated into the protocol is QoS function identifying the variation of similarity between two services of the same type, the current availability of each service, and how important such variation and availability is to the use.
3. A novel distributed constraint satisfaction problem model and protocol for virtual device composition in MANETs. The technique provides increased efficiency by taking a pull-based approach to service advertisement removing the need for constant periodic service advertisement by controlled broadcast, unnecessarily depleting node resources and congesting the network.
4. A capability reconciliation scheme resolving capability conflicts between composed services. Contributes a scheme for identifying and integrating network side media processing functionality, whereby the input of a service B is not compatible with the output of a service A; A and B needing to be composed.
5. A network performance influenced QoS model for the graceful degradation and upgradation of a virtual device post-composition. Contributes a method enabling a composed virtual device to adapt along with the conditions available in the user's environment.

1.5 Organization of the Thesis

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

Chapter 2 presents background necessary in understanding the virtual device problem. More specifically, various aspects and challenges of networked media are addressed in setting the tone for the remainder of the thesis.

Chapter 3 presents the state of the art of the virtual device and discusses various approaches adopted by the research community. It identifies limitations of various

research efforts for virtual device management in infrastructure-based and infrastructure-less pervasive environments.

Chapter 4 introduces the components and methodology behind a policy-driven autonomous framework for the dynamic discovery, selection, and composition of multimodal multi-device services. The framework operates in an ad hoc network setting and provided is a Service Overlay Network (SON) based definition of a virtual device.

Chapter 5 presents an extension to a prominent dynamic broker-based distributed service composition protocol, embedding in a distributed manner, a QoS model providing compositions that form the highest degree of match with regards to a user's desired task.

Chapter 6 describes a distributed constraint satisfaction problem for virtual device composition in MANETs addressing and resolving the issues faced by periodic service advertising by controlled broadcast.

Chapter 7 presents an extension to the distributed constraint satisfaction problem for capability reconciliation in MANETs, enabling the composition of conflicting services.

Chapter 8 produces a network performance influenced QoS model and its application for the graceful network-based adaptation of virtual devices post-composition in MANETs.

Finally, Chapter 9 summarizes the presented research work and discusses directions of future research work.

Chapter 2

Background

In this chapter we introduce various aspects and challenges of networked media, setting the tone for the remainder of the thesis. Discussed are the effects of mobility and the significance of mobility management. Addressed is the continued need for mobility management requiring cross-layer approaches. Identified and evaluated are various layer specific approaches. Moreover, established media delivery techniques are introduced and critiqued, placing emphasis on the development of adequate QoS models and, once again, cross-layer solutions for QoS adaptation. In an effort to overcome highlighted issues, aspects of the Ambient Networks Integrated Project [18] are brought forward and discussed in providing network side media processing for advanced media delivery. Lastly, a summary and discussion is provided setting the ideal of this dissertation with respect to current challenges in the realm of networked media.

2.1 Networked Media & User Mobility Management

Technological innovations with regards to media formats, wireless networks, terminal types, and capabilities have changed the way people work, live, and play. End users are now confronted with a wide variety of media services and applications providing access to *networked media*. Networked media can be defined as any kind of media including text, image, 3D graphics, audio, and video produced, distributed, shared, managed, and

consumed on-line through various networks like the Internet, Fiber, WiFi, WiMAX, GPRS, 3G, and so on, in a convergent manner [3]. With the pace of innovation being extremely high and no slow-down in sight, new models of interaction and cooperation are expected in the near future, able to support enhanced levels of quality of experience and novel on-the-move type applications including collaborative environments, personalized services, and on-line gaming. Such innovative models bring a number of challenges to multimedia delivery with regards to seamless multi-access connectivity enabling unfettered access to multimedia services regardless of location; service and content providers delivering their services to a broad variety of end-devices; and intelligent media routing. A common theme amongst these challenges is *user mobility management*.

2.1.1 Universal Mobility

Mentioned above is the fact that networked media travels through various networks in a convergent manner. In the last twenty years, the growth of wireless systems has become evolutionary. Generation by generation, wireless systems have grown to meet the needs of society, beginning with the first generation (1G) wireless systems, which have dwindled in importance, leading to the development of current and dominant generations such as 2G (e.g. Global System for Mobile Communications), 2.5G (e.g. General Packet Radio Services), and 3G (Universal Mobile Telecommunications System) wireless systems.

Wireless systems are not limited to Terrestrial Wide Area Cellular Systems. New and emerging wireless systems include Global Area Satellite Systems, Wireless Local Area Networks (WLAN), and Personal Area Networks (PAN). Today's wireless networks are quite heterogeneous, meaning they possess different ranges, capabilities, and purposes. However, they are complementary to each other in the way that if the variety of wireless networks were to be used as one large wireless network, users would become empowered as never before. However, this is where the challenge lies. The various wireless networks differ mainly within the lower layers (i.e. link and physical layers). This implies that the convergence of heterogeneous networks is plausible and

must be developed within the IP layers (i.e. network and application layers). This idea exists as the *all-IP vision*, in which network infrastructures and technologies are all-IP based. In this manner, solutions become compatible with, and independent from the actual access technologies; and hence, convergence.

Next-generation (NG) wireless systems (3G and beyond) are evolving towards a global vision described within the IMT-2000 initiative [20]. IMT-2000 is a general term for technologies that are to be included in the International Telecommunications Union's (ITU) world standard for NG mobile communication. ITU is an international organization, under the United Nations, concerned with telecommunications. One of the most important goals of the IMT-2000 initiative is universal mobility. *Universal mobility* can be defined as a user's ability to achieve "global connectivity and roaming, while having access to the same services and keeping consistent quality of service (QoS), anytime, anywhere, and using any terminal." [22] There are four different mobility aspects to universal mobility. They are terminal, session, personal, and service mobility.

- **Terminal mobility**: A user's ability to use their terminal to move across heterogeneous networks while having access to the same set of subscribed services. This includes both *micro-mobility* and *macro-mobility*, whereby macro-mobility is the movement of a mobile user between two network domains, and micro-mobility is the movement of a mobile user between two subnets of one network domain. Terminal mobility includes *handoff management*, which is the act of keeping an ongoing call alive as the mobile host changes its point of attachment, and *location management*, which is the act of tracking and locating an idle mobile host.
- **Session mobility**: A user's ability to maintain an active session while switching terminals.
- **Personal mobility**: A user's ability to be globally reachable by a unique personal identifier and originate or receive a session on any authorized terminals.
- **Service mobility**: A user's ability to acquire their personalized services, with an expected QoS, regardless of their location.

The following scenario is presented to help better understand the aspects of universal mobility.

In the morning, on his way to work by train, a mobile user (m_user), using his PDA, engages in a video conference. Although the train traverses several sub-networks of an administrative domain, m_user does not notice any disruptions (terminal mobility). Reaching his office from the outdoors, m_user transfers the video conference session to his desktop PC (session mobility). Later on in the day, m_user leaves work for a meeting at a conference center, where he accesses a desktop PC and receives an invitation from a colleague to join another video conference. The colleague, unaware of m_user's mobility and current location, establishes communication with m_user via m_user's working address (personal mobility). In the evening, m_user flies to another country and roams into a cellular system that is served by another network service provider. However, m_user can still receive his personalized services (service mobility). (Adapted from [22]).

It was stated above that through an all-IP vision networks shall converge, where mobility functions will be developed within the IP layers of the TCP/IP protocol stack. Again, the reason for this being that the lower layers of the TCP/IP stack are where the main differences lie between the various heterogeneous networks. Therefore, the lowest possible layer to provide mobility functionality is the network layer, which we now examine.

2.1.2 Mobility Management at the Network Layer

Mobility management schemes at the network layer can be divided into two groups: those supporting macro-mobility and those supporting micro-mobility. Macro-mobility is by far the more complex of the two groups as it handles mobility at a global level, whereas micro-mobility implies mobility at a more sub-local level. At the network level, macro-mobility can be handled through the use of Mobile IP (MIP) [23, 24]. MIP is a proposed standard for mobility management developed by the Internet Engineering Task Force (IETF), growing from version 4 (IPv4) to version 6 (IPv6). MIP allows a mobile host to keep a permanent static IP address as it moves from its home network into a variety of foreign networks. Of course this is simply an illusion, as it is impossible due to addressing schemes used throughout the Internet. However, MIP successfully

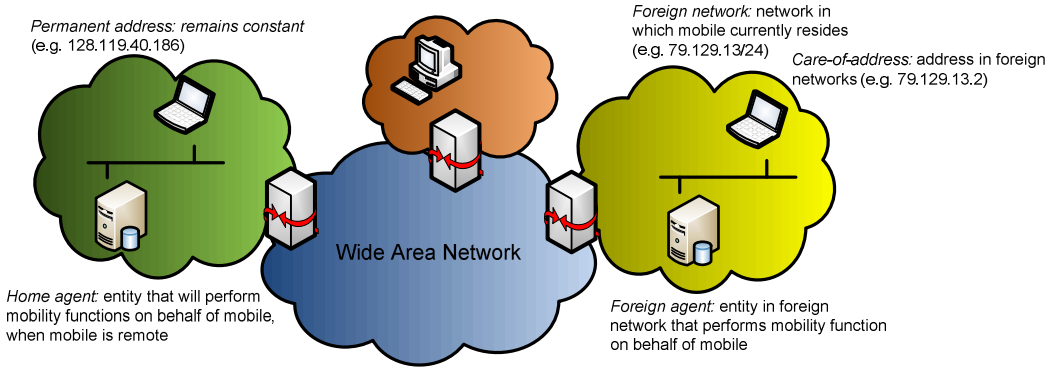


Fig. 2.1. Mobile IP component overview.

provides this service through the use of agent entities and tunneling. MIP has the ability to handle micro-mobility, but does so in an inefficient manner; therefore, MIP is more efficient when accompanied by a micro-mobility extension (e.g. IDMP [25], Cellular IP [26], and HAWAII [27]).

MIP introduces three functional entities: a home agent (HA), a foreign agent (FA), and a mobile node (MN). MIP allows a corresponding node to communicate with a MN using the home address of the MN (i.e. permanent address), even as the MN moves into foreign networks and acquires new IP addresses (i.e. care-of-addresses). The method revolves around the idea that the MN is represented by a home agent entity in its network of origin (i.e. home network) and a foreign agent in a visited network (i.e. foreign network). It is the job of the home agent to intercept any packets addressed to the permanent address of the MN and forward them to the foreign agent using the care-of-address (CoA), who will then pass sent packets to the MN, all in a transparent fashion. The MN will then respond to the correspondent host using its corresponding address, where the correspondent could be mobile itself (see Figure 2.1). The first step to this process is agent discovery.

Agent discovery is performed when a MN enters a foreign network. The MN must discover (i.e. agent solicitation), or be discovered by (i.e. agent advertisement) a foreign agent. The process of discovering an agent is how a MN detects its movement into a new subnet. As just reflected, there are two methods of agent discovery: Agent Advertisement, where the MN receives periodic unsolicited broadcasts from FAs, and Agent Solicitation, where the MN itself sends messages in order to learn about any

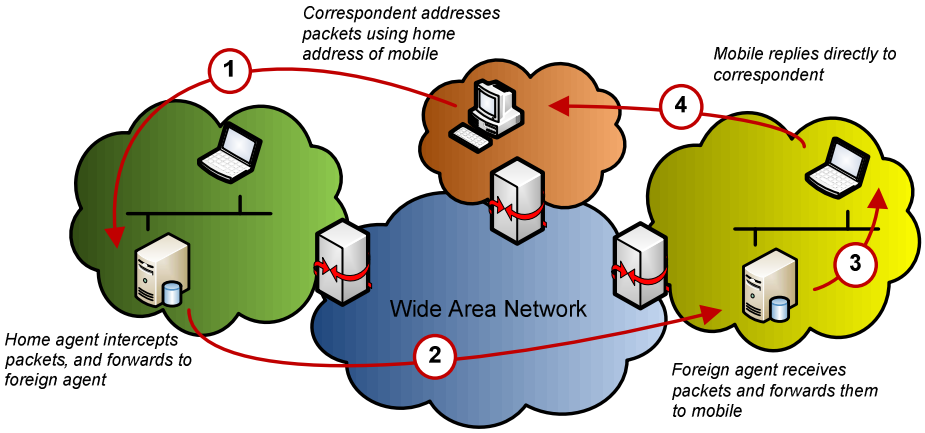


Fig. 2.2. Mobile IP routing.

prospective FAs. After the discovery of an agent, registration with the home agent is required.

In the process of discovering a foreign agent the MN has acquired a new CoA. The HA must keep a link associated with the MN's permanent IP address and its CoA, which is referred to as a binding. When the MN accepts a new CoA, the HA must be informed and perform a binding update, which successfully completes registration. Upon registration, packets can now be correctly routed to the MN.

At this point, the correspondent entity will address any packets sent to the MN to the MN's permanent address, where the correspondent is completely oblivious of the MN's mobility. The HA intercepts sent packets and encapsulates them using IP-within-IP [28], where the outer IP contains the CoA address and the inner IP contains the permanent address of the MN. Sent packets are then tunneled to the FA serving the MN, at which point they are decapsulated and forwarded to the MN. Using the home address of the correspondent, the MN is able to communicate directly with the correspondent, where the MN uses its permanent address as the source address (see Figure 2.2).

Even with Mobile IP and a variety of micro-mobility extensions, all the network layer has to offer towards supporting universal mobility is strong terminal mobility. The network layer excels at low- level mobility (i.e. terminal mobility), but fails miserably at high level mobility (i.e. session, personal, and service mobility). Mobile IP, with a micro-mobility extension, handles terminal mobility efficiently on both the macro and

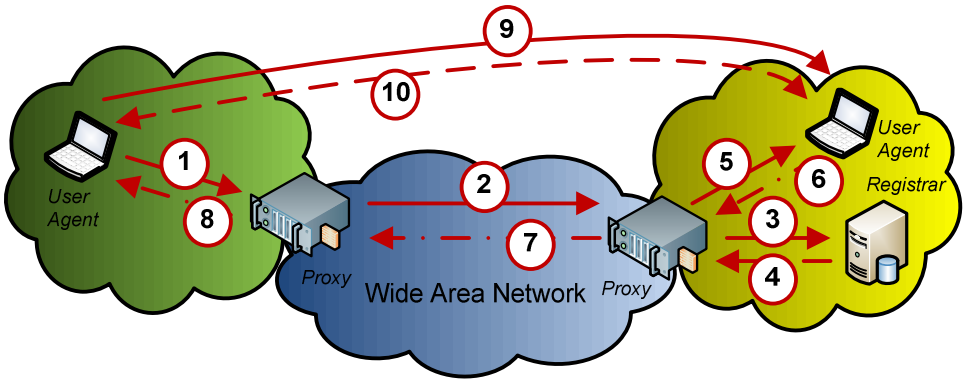


Fig. 2.3. Example SIP message flow.

micro levels. As a mobile node moves from one domain to another, the mobile node will obtain a new care of address, which the home agent updates as the new binding upon re-registration with a new foreign agent. This act establishes location management and completes the handoff. The home agent encapsulates and routes intercepted packets to the bound care-of-address, hence allowing for consistent information to flow seamlessly as the mobile node moves. This consistent flow of information is extended to subnet mobility as micro-mobility extensions correctly route the decapsulated packets to the mobile node based on the extension's routing mechanism.

Having looked at mobility management in the network layer, we now consider the application layer.

2.1.3 Mobility Management at the Application Layer

A popular approach to mobility management within the application layer is the Session Initiation Protocol (SIP) [29, 30]. SIP, an IETF signaling protocol, is quickly gaining widespread acceptance as the signaling protocol of choice for Internet multimedia and telephony services. As part of its signaling mechanism, SIP already supports personal mobility, and furthermore, SIP's feature set can easily be extended to support adequate means of terminal, session, and service mobility.

SIP is a textual client-server protocol, where the client issues a request and the server returns a response. SIP provides the means to establish, maintain, and terminate IP multimedia sessions. SIP does not transport the media itself, but rather provides the

signaling and media description necessary for another protocol (e.g. RTP) to handle the task. Figure 2.3 depicts an example sequence of requests and responses in acquiring a session between two users. SIP is composed of user agents and network servers. User agents take on the roles of *User Agent Client* (UAC) or *User Agent Server* (UAS), and network servers take on the roles of proxy, redirect, or registrar. User agents represent the user, whereby UACs initiate a SIP request and UASs receive a SIP request and return a SIP response. The communication between user agents is accomplished either directly or through the use of SIP network servers (i.e. proxy, redirect, and registrar). *SIP proxy servers* provide the correct forwarding of incoming SIP requests by receiving a SIP request, determining the next hop, and then forwarding the request. SIP proxy servers have the option of being stateless or stateful. A stateful SIP proxy server, unlike a stateless one, is able to maintain all information related to a SIP transaction (i.e. a collection of message exchanges), which allows proxy servers to contribute additional functionality such as the handling of personal mobility. A *SIP redirect server's* only task is to receive a request and instruct the client to contact the next hop directly. *SIP registrars* are key components of SIP that act as the registration authorities within SIP domains and that are responsible for user location management.

The entities addressed in SIP are physical users, which are identified by SIP uniform resource identifiers (URI). A SIP URI is equivalent to an email address in the way that it takes the same form: "SIP:user@domain", where 'user' is a user name, telephone number, or civil name, and 'domain' is a domain name or numeric address. All user agents are required to register with a local registrar. The location server, which is part of the registrar, is responsible for maintaining user locations. It does so by binding an IP address given by a user agent during registration to the SIP URI of the user. In this manner, a user can be located through the act of querying a registrar, which is done by SIP proxy and redirect servers. Since there may be more than one IP address bound to a single SIP URI, a list of one or more locations where the user may be found is returned to the server as a result of the query.

There are two cases to locating a user. Case one refers to the user being in the domain served by the proxy or redirect. In this case, the proxy or redirect contacts the location server for the registration information of the corresponding address and proxies

or redirects the request as needed. The second case refers to the user not being within the domain served by the proxy or redirect. In this second case, the proxy or redirect is able to use the domain name server (DNS) to proxy the request further.

SIP is a very flexible protocol and new services and functionalities can easily be supported through SIP extensions. A few popular extensions of SIP include presence [31], event state publication [32], third-party call control [33], and the REFER method [34].

In basic SIP, users must pull other users into a session, such that a user must query other users as to their availability and ability to participate in a session. Presence is more of a push, as users are made aware of other users. This is achieved through an entity known as a presence agent (PA), which stores information regarding the presence of a user. A user must subscribe to the PA and publish their presence information whenever it changes. The PUBLISH and SUBSCRIBE methods are also extensions to SIP, which allow an entity to record its state and state changes, such as presence.

Third-party call control (3PCC) is an extension of SIP that enables a SIP agent to act as a controller, which can both initiate and manage sessions between two users. 3PCC is useful in conferencing applications and operator services. More specifically, 3PCC can be used to achieve session mobility (discussed in Section 2.1.2.2). A second extension that can be used in session mobility is the REFER method, which enables a user to transfer a session to another terminal, whereby a REFER message instructs the receiver to INVITE a new terminal based on contact information contained within the message.

SIP, being an application layer approach and providing an all IP-based solution, has significant potential for supporting mobility management as well as the convergence of heterogeneous networks. The following examines how SIP achieves support for the four aspects of universal mobility.

2.1.3.1 SIP and Terminal Mobility

SIP handles terminal mobility at two stages: pre-call and mid-call. Pre-call mobility represents a mobile host entering a new network domain prior to the existence of a media session. In this case, pre-call mobility triggers a mobile host to re-register with its

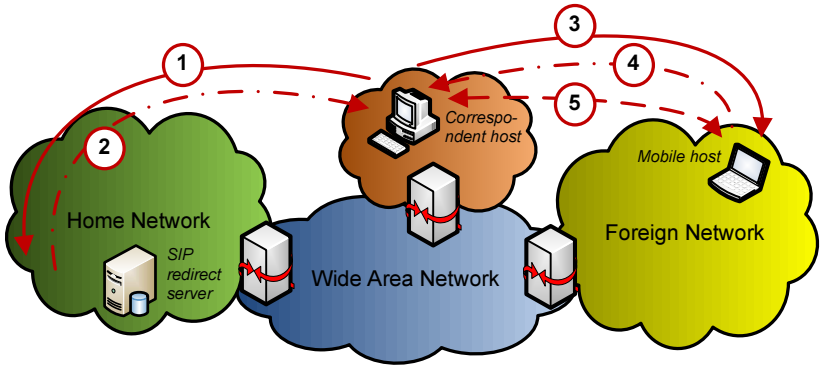


Fig. 2.4. SIP and terminal mobility (pre-call).

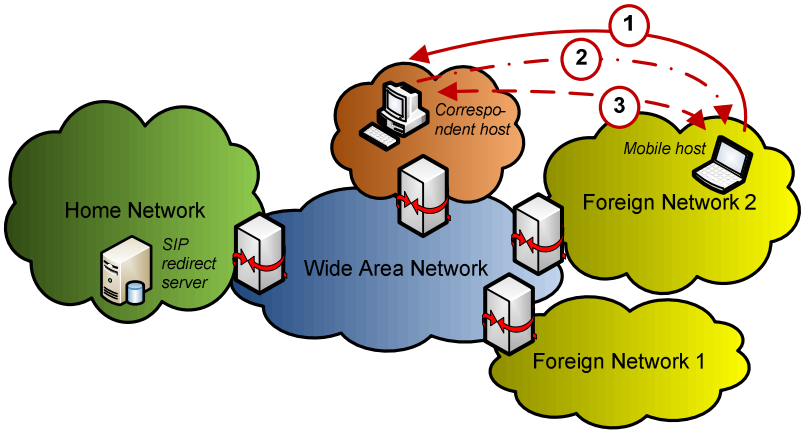


Fig. 2.5. SIP and terminal mobility (mid-call).

home registrar, as upon entering a new domain the mobile node's IP address has changed. This action signifies a location update, such that the registrar contains an up-to-date listing pertaining to the mobile host's current location (i.e. IP address). Moreover, this enables a registrar to redirect any incoming request bearing the SIP URL of the mobile host to the new location (actions 1-5 in Figure 2.4).

Mid-call mobility represents a mobile host entering a new network domain while carrying an active media session. In this case, the mobile host is required to re-INVITE the corresponding host (actions 1-3 in Figure 2.5), where the invite contains an updated session description with a new IP address.

SIP, as an application layer approach, is very well equipped to handle high level mobility issues, but struggles in terms of efficiency to handle low level mobility issues. Since terminal mobility is a low level mobility issue, SIP provides a less efficient

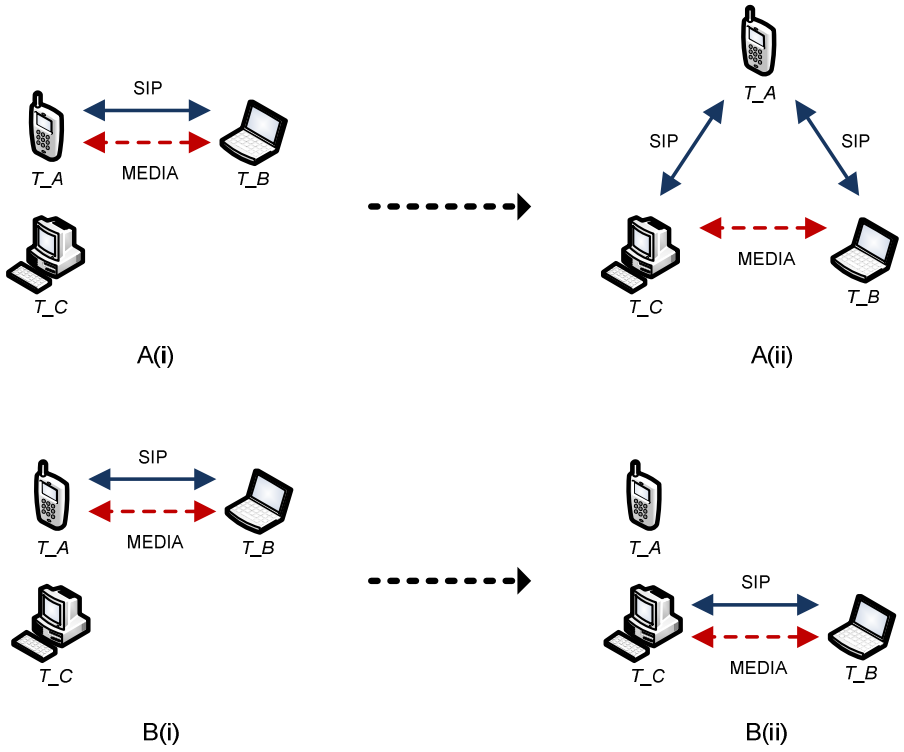


Fig. 2.6. SIP and session mobility version A and B, before (i) and after (ii).

solution, due to the fact that it requires a SIP client, representing the mobile host, to poll the operating system every few seconds to notice a change in network domain.

2.1.3.2 SIP and Session Mobility

Session mobility is a user's ability to maintain an active session while switching terminals. SIP handles session mobility by redirecting the session between two parties using either the Third party call control (3PCC) extension or the REQUEST method extension. Using 3PCC, a terminal_A wanting to switch its session to a new terminal_C, first establishes a SIP session with terminal_C. Second, terminal_A instructs its current communication partner, terminal_B, to redirect its media stream to terminal_C. When this is completed, terminal_A has successfully moved its media session to terminal_C. This is depicted in Figure 2.6 A(i) and A(ii).

Using the REQUEST method extension, terminal_A switches its session to terminal_C by sending a REQUEST message to terminal_B instructing it to establish a

session with terminal_C. At this point, terminal_A terminates its session with terminal_B. When this is completed, terminal_A, once again, has successfully moved its media session to terminal_C. This is depicted in Figure 2.6 B(i) and B(ii).

2.1.3.3 SIP and Personal Mobility

Personal mobility is the user's ability to be globally reachable by a unique personal ID and originate or receive a session on any authorized terminals. Through the use of registrar and location servers, SIP offers transparent mapping and redirection services. As mentioned previously, the relation between SIP URIs and user agents (i.e. terminals) is one-to-many, and therefore, the registered IP addresses of all terminals associated with a user are kept and made available at any given time. This enables the proper redirection of calls to the appropriate terminal.

In the case of a single SIP URI being associated with multiple devices belonging to one user, a call forking procedure is used within a SIP proxy server. This procedure forks a call to all the active terminals associated with a specific SIP URI. Each UAS, representing a terminal, sends a response to the proxy, at which point, the proxy makes use of the implicit ordering of SIP responses to forward the best response back to the UAC. In this manner, a call is setup with the most appropriate terminal.

2.1.3.4 SIP and Service Mobility

Service mobility is the ability of the network to provide personalized services to the user, with expected QoS, regardless of the user's location. A simple solution to personal mobility is to have a user's service information present on their actual terminal. In this case, the user carries their service information as they carry their terminal. However, the record of their service information is limited to that terminal, which does not account for the need for service information to be present when using any other authorized terminal or multiple authorized terminals. SIP provides a solution to this dilemma by including the service information of a user as part of the registration process, whereby each time a terminal is registered, the most recent version of the user's service information is used.

Having examined two mobility management solutions at the network and application layers, we now compare them from a theoretical perspective.

2.1.4 A Theoretical Comparison of Mobility Management Approaches

The Mobility management approaches shall now be discussed in a comparative fashion based on the following important parameters related to mobility support.

- The ability to support real-time traffic in the presence of mobility.
- The ability to support non-real-time traffic, specifically consistent TCP connections, in the presence of mobility.
- The ability to keep a good level of transparency, in the presence of mobility.
- The ability for both communicating parties to be simultaneously mobile (i.e. dual movement).

Comparison	Network Layer (MIP)	Application Layer (SIP)	Optimal Layer
Real-time traffic support	no	yes	application
Non-real-time traffic support (consistent TCP connection)	yes	no	network
Transparency level	medium to high	low	network
Dual movement	yes	no	network
Universal mobility support	no	yes (with weakness)	neither
High-level mobility support	no	yes (strong)	application
Low-level mobility support	yes (strong)	yes (weak)	network
Terminal mobility support	yes (strong)	yes (weak)	network
Session mobility support	no	yes (strong)	application
Personal mobility support	no	yes (strong)	application
Service mobility support	no	yes (strong)	application

Table 2.1. Theoretical comparison summary

- The ability to support complete universal mobility.
- The ability to support high-level mobility (i.e. session, personal, and service mobility).
- The ability to support low-level mobility (i.e. terminal mobility).

Table 2.1 summarizes the findings of the comparison. An ideal mobility management scheme needs to support both real-time (e.g. real-time multimedia) and non-real-time traffic (e.g. consistent TCP connection) in the presence of mobility. MIP provides excellent non-real-time traffic support, as two parties are able to keep an ongoing TCP connection while completely oblivious of existing mobility. However, MIP, which was developed at a time when real-time traffic was not a concern, lacks the proper definitions in terms of QoS requirements. In contrast, SIP is designed specifically for the handling of real-time traffic, and so lacks the ability to keep a consistent TCP connection alive, as its handoff scheme requires the breaking of a TCP connection. However, SIP is capable of supporting TCP communication as long as the duration of the TCP connection is very short.

MIP has a significant advantage over SIP when handling low-level mobility (i.e. terminal mobility), due to fact that MIP is located at the network layer. This makes sensing a change in underlying network or sub-network much more efficient than can be achieved by SIP. Another advantage of being located at the network layer is the ability to hide the process of terminal mobility from the application layer.

In regards to high-level mobility (i.e. session, personal, service), the application layer dominates. SIP, with its ability to be integrated with a variety of protocols, is an extremely flexible and powerful tool in meeting higher level mobility. This is something that the network layer presently cannot achieve. Through the use of a series of requests and replies, SIP is able to bring the idea of a session to a higher and independent level, allowing sessions to be transferred and split across multiple terminals. Furthermore, through its addressing scheme, registration technique, and forking mechanism, SIP is able to provide personal and service mobility, allowing users to be represented by a unique global identifier and keep their personal services with them, regardless of their location or what terminals they use.

Table 2.1 shows that support for universal mobility is not a single layer task. The responsibilities of universal mobility are almost divided in an even fashion between the network and application layers, where the network layer is well suited for low-level mobility and the application layer is well suited for high-level mobility. This observation is a strong clue that a complete solution to universal mobility will more than likely originate as a hybrid of multiple layers. The literature has presented some attempts to combine the application and network layers into a multilayered approach to support universal mobility. Politis et al [35] propose a multilayered solution, which uses a combination of MIP and SIP, such that SIP is responsible for real-time traffic and MIP is responsible for non-real-time traffic. This scheme possesses the advantages of both SIP and MIP, but introduces new complications, such as signaling problems, due to the fact that a mobile node is required to register upon entering a foreign network with both its home agent (required in MIP) and home registrar (required in SIP). This results in redundant functionality and data repetition. Furthermore, dual movement is still an issue when using SIP. Therefore, it can be said that multilayered approaches are not yet feasible and quite premature in their development, but shall eventually appear as the more optimal mobility management solutions.

Having stressed the importance of mobility management, we now focus on current media delivery standards and Quality of Service (QoS)

2.2 Multimedia Delivery Standards and QoS

Although many solutions exist for providing multimedia services, only a few are regarded as standards for multimedia delivery. These include the Session Initiation Protocol (SIP), the IP Multimedia Subsystem (IMS), and Content Distribution Networks (CDNs). This section begins with a discussion of these technologies in identifying their shortcomings. Subsequently, QoS with respect to multimedia is introduced such that sustainable end-to-end quality is still one of the main challenges in today’s networked media.

2.2.1 Session Initiation Protocol (SIP)

As introduced in the previous section, SIP is an application layer control protocol for establishing, maintaining, and terminating media sessions between one or more participants; particularly real-time communications. SIP has proven to be a very flexible and powerful technology; however, it suffers from a few setbacks. SIP was designed to support point-to-point communication, with end-to-end usage as a priority. Such a design forces media adaptation to occur at the end devices and does not provide for efficient support for multi-point communication. Handling such issues requires the addition of non-standard approaches. Moreover, another setback for SIP is the lack of support for peer-to-peer. Furthermore, in terms of mobility support, although SIP does handle various degrees of mobility, being located at the application layer makes for long handover periods and the inability to handle the effects of network load and failures.

2.2.2 IP Multimedia Subsystem (IMS)

IMS is defined by the 3rd Generation Partnership Projects (3GPP and 3GPP2) and is the first platform standardized in the direction of network-independent access and session control [36, 37]. IMS is a service overlay architecture enabling the provision of highly integrated multimedia services on top of various networked technologies. Moreover IMS is entirely built on Internet protocols defined by the Internet Engineering Task Force (IETF) and is based on SIP for multimedia session establishment, modification, and termination. Being based on SIP implies that IMS also inherits SIP's shortcomings; however, improvements have been integrated to better handle broadcast communications and low-level mobility. Unfortunately, current versions of IMS still rely on SIP-based techniques for mid-session macro-handover, causing unacceptable delays for delay-sensitive real-time services [38]. Also, similar to SIP is the lack of dedicated adaptation components forcing media adaptation to occur at end devices.

2.2.3 Content Distribution Networks (CDNs)

CDNs provide an approach to media delivery whereby multimedia content load is distributed among many sites throughout the network [39, 40]. Through the use of multicasting protocols and caching technologies providing integrated distribution, streaming, security, and traffic management solutions, high performance delivery can be achieved. However, the concepts behind CDNs were established for the sole purpose of efficient media data transport, and neglect the intelligence required for media adaptation, user mobility, and network load balancing.

QoS plays a very important role in media delivery, such that variations in QoS are extremely obvious to the user and dramatically affect overall user experience.

2.2.4 Quality of Service in Media Delivery

Multimedia is evolving how QoS is defined, measured and evaluated. Although many definitions for Quality of Service exist, we refer to the International Telecommunication Union's (ITU) definition of QoS [41], which is "the collective effect of service performances which determine the degree of satisfaction of a user of the service." Like most general definitions for QoS, this definition is very broad. When it comes to media delivery, QoS is all about the end user's perceptual experience, which require both *objective* and *subjective* methods of measurement and evaluation [4]. An example of an

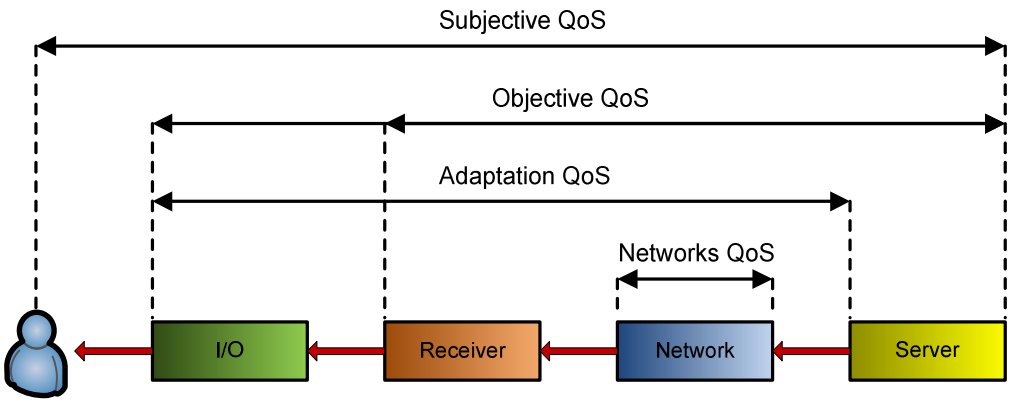


Fig. 2.7. Multimedia QoS Abstraction

objective method is the analysis of signals in both compressed (e.g. MPEG-4/H.264) and non-compressed (e.g. reconstructed RGB video) formats. Subjective methods, on the other hand, attempt to obtain the user's perceived QoS and are much more complex as they involve a large number of tests operated under controlled psychometric experimental conditions to obtain meaningful results.

Overall and at a high level, QoS can be decomposed into a series of overlapping sub-QoS categories (see Figure 2.7): *subjective QoS*, *objective QoS*, *adaptation QoS* (AQoS), and *network QoS* (NQoS) [4]. At the lowest layer of the spectrum is NQoS, representing the more traditional view of QoS, incorporating aspect such as bandwidth, delay, jitter, and packet loss rate. AQoS refers to the quality of the adaptation of media to accommodate the heterogeneous aspects of both terminals and networks. Objective QoS reflects the composed end result of AQoS and NQoS, and subjective QoS introduces the user's perception.

From a service point of view, and building on the above, at the highest level is the user's *Expected PQoS*, representing what the user expects from a particular contracted service, based on the user's preferences and the capabilities/constraints of their utilized terminal. As the service is provided by an application, and every application has performance needs and constraints such as encoding/transmission parameters, frame rate, resolution, and etc., an *Application QoS* (ApQoS) is introduced. Upon service consumption a *Delivered PQoS* is established, whereby the ideal is to select an application to deliver the service with an ApQoS best fitting the Expected PQoS to achieve the best possible Delivered PQoS.

Implementing the above QoS model is far from being achieved and is proving difficult. To understand why, we examine the current Internet architecture.

2.2.4.1 Quality of Service and the Current Internet Architecture

The Internet architecture of today is based on the clean modular structure of TCP/IP and UDP/IP models, whereby their layers are implemented in complete isolation of each other. With the inception of heterogeneous networks, terminals, and real time delay sensitive applications, the desire to optimize these protocols and their respective

mechanisms is strong, in order to improve QoS, increase throughput, and efficiently utilize bandwidth [4].

As mentioned, each layer associated with TCP/IP and UDP/IP models is independent of the next, and requires a distinct set of mechanisms and respective parameters to fulfill its functionality. Such information is classified by layer in Table 2.2. Looking at the table one can notice that unlike the traditional single layer operation of TCP/IP and UDP/IP, achieving optimal QoS shall require a cross-layer approach promoting layer interaction and a global optimization technique; a problem which is proving illusively challenging [42]. Although efforts have been made with the introduction of MPEG-21-

Layers	***Mechanisms (to optimize)***	***Parameters***
User	User priority selection	Terminal characteristics, objective quality metrics (e.g. PSNR, VQM, SSIM), Subjective quality metrics (e.g. MOS, SAMVIQ, DSIS)
Application	Transrating, Transcoding, Forward Error Correction (FEC), Automatic Repeat Request (ARQ), Adaptive encoding/decoding	Rate, Codec, Protection level
Transport	TCP Congestion Control, UDP, Header Compression	Packet loss information, receiver window, congestion window, retransmission timer
Network	Packetization, DiffServ, TE	IP packet size, DiffServ Code Point, Handoff information
Data Link	MAC Protocols, Radio resource control, FEC, ARQ, Framing	Retransmission attempts, Error rate, retry limit, RTS/CTS, Handoff, Traffic classes, TDMA time slots, OFDM carriers
Physical	Channel modulation and coding	BER, signal strength, transmission power, capability profile
Context Information	Dynamic Voltage Scaling, Scheduling	Battery status, Architectural capability profile

Table 2.2. Mechanisms and Parameters at Different Layers (adapted from [4]).

enabled cross-layer adaptation [43], many challenges still remain in dealing with issues such as how to handle the lack of clean isolation between layers; how to optimize cross-layer parameters; how and where to control cross-layer adaptation; and how to assure fairness.

Having discussed the various aspects of networked media in terms of mobility management, delivery standards, and QoS, we now present a solution to media routing and transportation originating from the Ambient Networks Integrated Project [18]. The presented approach attempts to handle many of the missing functionalities and setbacks discussed in this chapter.

2.3 Smart Media Routing and Transport (SMART)

The Ambient Networks Integrated Project [18] sets the foundation to a vision for the future of wireless and mobile networks. The focus of this work builds on the All-IP vision and the trend of structuring the Internet into smaller domains. The project provides a novel networking concept that enables the cooperation of heterogeneous networks belonging to different operator domains. Moreover, at the root of the project is a common control layer for various network types, providing flexible and dynamic network configuration, able to supply end users with seamless multi-access connectivity ensuring the best possible network connection at all times. Amalgamated into this vision are broad amounts of heterogeneity in terms of access networks, terminals, network interfaces, users, signaling and transport protocols, applications, and services making media routing and transport a significant aspect of the project. Reason being that the large spectrum of heterogeneity results in multimedia data needing to be proactively cached, trans-coded, split, synchronized, translated, filtered, or transformed in some way or another before being delivered. As previously discussed, existing multimedia delivery standards leave much to be improved in this area, and hence to this end, the Smart Media Routing and Transport architecture [19] has been proposed to enable the seamless integration of next-generation multimedia services into Ambient Networks.

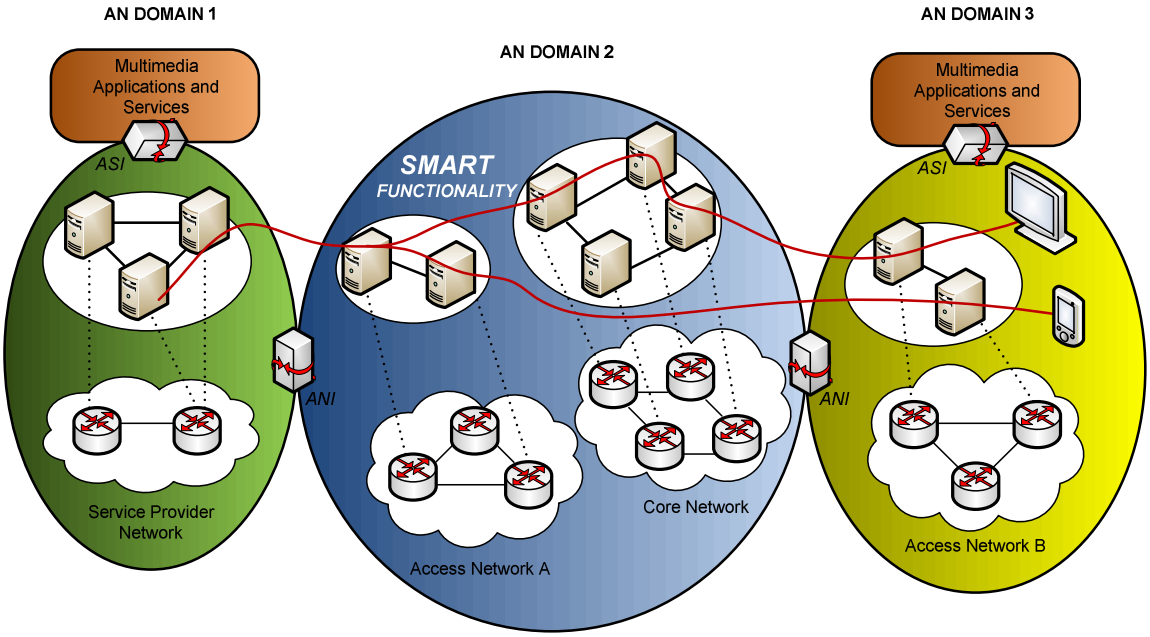


Fig. 2.8. SMART Functionality in AN Scenario mobility management mechanisms, for example.

2.3.1 SMART Overview

Figure 2.8 illustrates where SMART functionality fits into the overall Ambient Network. Shown in the Figure is the delivery of a service, which can range from simple requests for information (e.g. web browsing) to multimedia streaming and conferencing, to more complex service scenarios including media routing and media adaptation to deal with terminal and user mobility; media splitting to enable session/flow mobility; synchronization for re-combining split flows; and smart caching for accommodating low bandwidth access networks. Such features can only be processed along the media path, and thus, inside the network, as opposed to server-side or client-side.

In SMART, multimedia transformation is carried out by network-side media processing capabilities and transformation services [44], termed *MediaPorts* (or MPs), which are located somewhere on the media path, between the sink, called *MediaClient* (or MC) and the source, called *MediaServer* (or MS). MPs must be able to transform the multimedia data originating from the MS into a form that is acceptable for the MC.

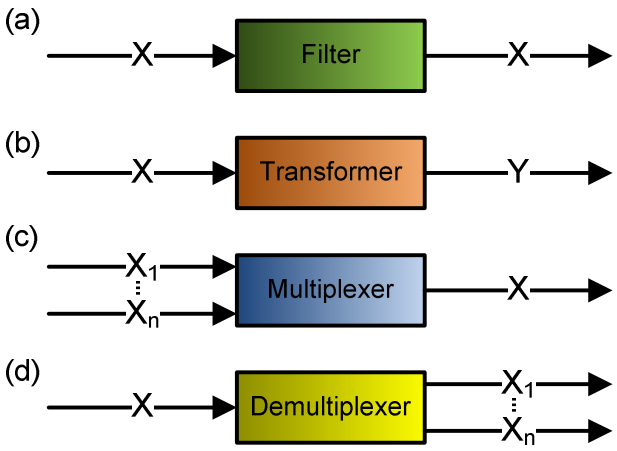


Fig. 2.9. Atomic Media Processing Classes

MediaPorts are an abstraction of the underlying network topology and are interconnected across a composed Ambient Network by the common control layer, which is facilitated via the *Ambient Network Interface* (ANI). Service/content providers define media services in the form of service profiles identifying requirements such as end-to-end QoS requirements and needed networks-side functions, which is facilitated via the *Ambient Network Service Interface* (ASI).

The main challenge addressed by SMART is the ability to cope with the heterogeneity of network technologies and user terminals, and to support the rapidly changing communication environment occurring from increasing numbers of mobile users and foreseen moving networks. Achieving such a task requires the tight integration of the media routing and transport architectures with other control functions of the Ambient Network, including the underlying connectivity, context management, and

2.3.2 Media Processing Functionality

Actual media processing services can be decomposed into the composition of four atomic classes shown in Figure 2.9, connected in serial or parallel [45]. Each of these classes serves a particular function and are introduced bellow.

When a *filter class* is applied to a media stream, it does not change the way the media is interpreted, but rather it simply performs some operation such as integrity

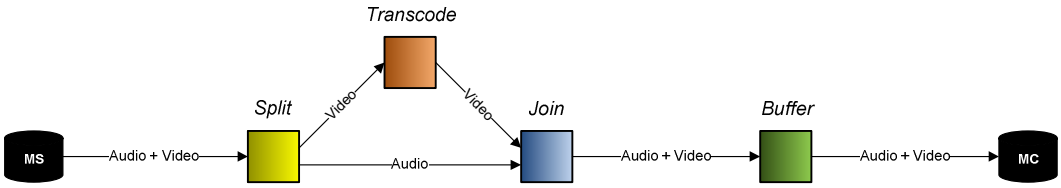


Fig. 2.10. Sample Media Adaptation

checking or buffering. On the other hand when a *transformer class* is applied to a media stream, it alters the nature of the media, such as transcoding the media stream to a different codec. A *multiplexer class'* role is to take several streams and join them into a single complete stream, such as a synchronization operation executed on a video and audio stream. Vice versa, a *demultiplexer class* splits a complete single stream into its component parts, such as a single stream containing audio and video being split into two streams; one being audio and the other being video.

Provided in figure 2.10 is a sample media stream being adapted from Media Source to MediaClient. The adaptation sequence depicts a scenario whereby a single media stream containing audio and video is sent from the MediaSource. In order for the MediaClient to correctly view the media, the video codec must be adapted to one that it supports. Therefore, the stream is first split into its component parts, at which point the video can be transcoded, rejoined with the audio and then proceed to be buffered for consumption by the MediaClient.

2.3.3 SMART Architecture

The functionality discussed above is provided by SMART using the key concept of *Service-Specific Overlay Networks* (SSON). A SSON is a overlay network and can be defined as a set of overlay nodes and links forming a virtual network. Such network entities, termed *Overlay Nodes* (ONodes) represent one of the main components of the SMART architecture (depicted in Figure 2.11) and enable the provisioning of the media processing capabilities discussed above. At any given time an ONode can be providing several media processing capabilities and be part of multiple SSONs.

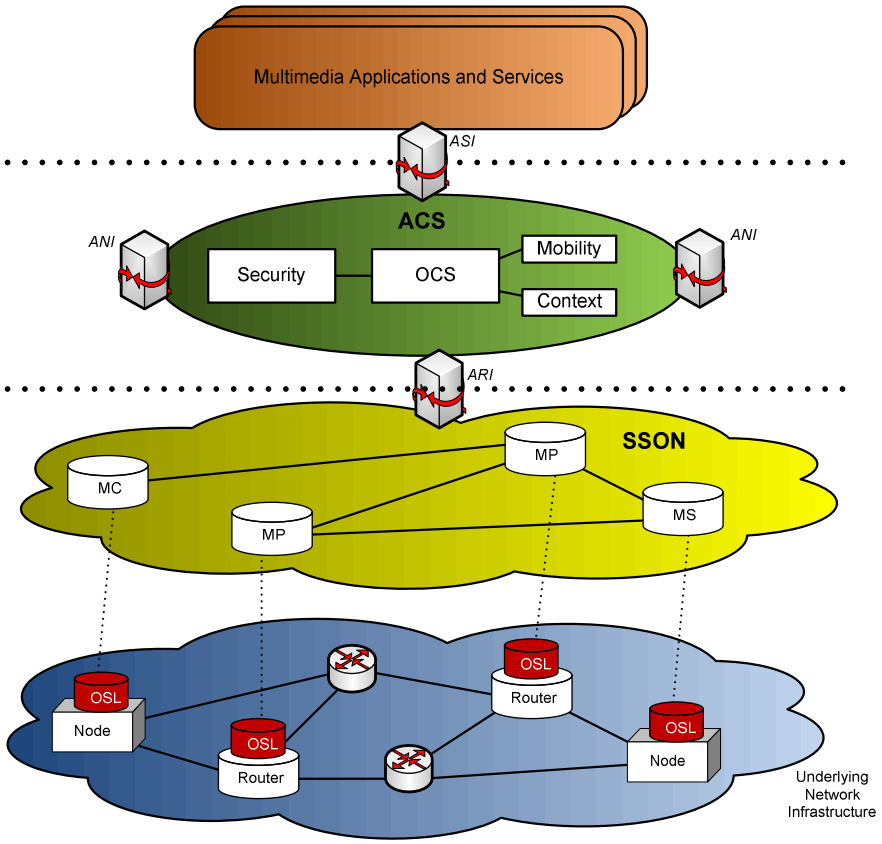


Fig. 2.11. SMART Architecture

A SSON (i.e. virtual network) is deployed for every media delivery service allowing for the configuration of appropriate overlay-level routing paths meeting the exact requirements of a particular service. Moreover, this permits the transparent integration of network-side media processing, and provides fine-grained adaptation and flexibility on a per-flow basis.

The SMART architecture deploys two main functionalities into the system. The first deals with the intelligent selection and configuration of ONodes to be included in a SSON, based on the specific needs of the service and the user. The second deals with the transportation and handling of user-data present in a SSON to carry out media processing operations. These two functionalities are respectively grouped in the *Control Plane* and *User Plane* of an ONode (depicted in Figure 2.12).

The control plane is responsible for the general management of ONodes and signaling exchange. Included in the control plane is the *ONode Control Entity*, which consists of several components separated into two classes; those that logically belong to

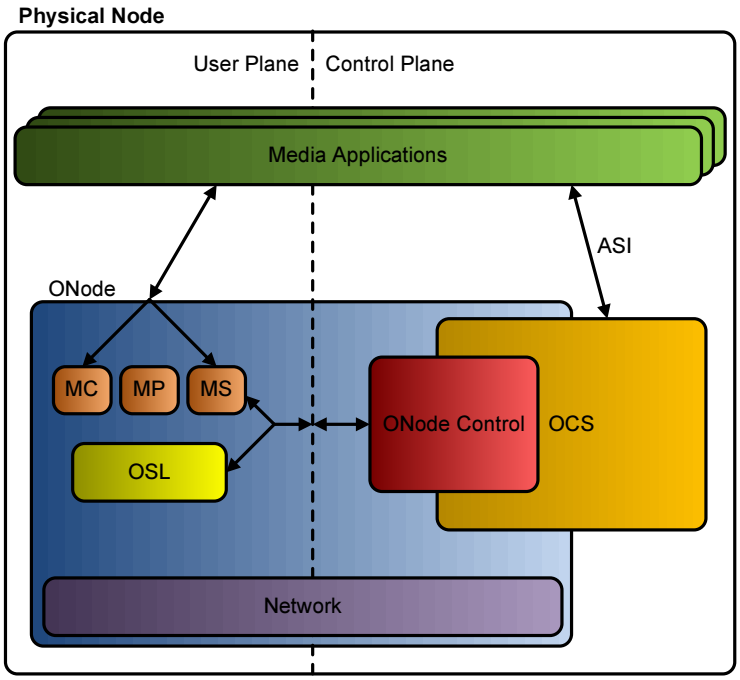


Fig. 2.12. ONode implementation

the *Overlay Control Space*, and those that deal only with local control and management. Components fitting within the first class constitute the *Media Delivery Functional Area* of the *Ambient Control Space*, which controls SSONs on an Ambient Network wide basis.

The *Overlay Support Layer* (OSL) and the application modules taking part in media processing actions are contained in the user plane of the ONode. Sitting on top of the underlying network, OSL embodies the basic overlay network functionality necessary for the handling of packets at the overlay level and provides a common communication abstraction to all ONodes of a SSON, allowing them to communicate with each other independent of actual differences in terms of underlying protocol stack and technologies. Immediately above OSL are application modules implementing MC, MS, or MP behavior.

2.3.4 SMART Use Case

User mobility shall play a key role in future office environments, such that ongoing sessions such as conference calls or incoming VoIP connections will always be routed to the most suitable end devices. As a result, media streams need to be adapted to the capabilities of respective end devices. Moreover, the splitting of media into distinct audio and video flows might be necessary, as previously seen, so that a user could see his dialog partner's video on their handheld, while listening to the audio routed directly to a Bluetooth headset. In addition, should a user be unavailable for a moment, networked-based caching of data targeted to the user would be a convenient service.

All of these described services require active network support, which can be provided by the SMART architecture. All communication takes place between MSs and MCs with MPs in between providing the respective media adaptations or other processing of the media stream. For example, when a session is transferred, as above, to a PDA and Bluetooth headset, a series of adjustments need to be made. Firstly, an adjustment to the PDA's small screen size and lower communication bandwidth must take place, involving a new ONode with transcoding capabilities to be integrated into the derived SSON. Moreover, a second ONode splitting the incoming stream into to distinct video/audio streams is necessary. All synchronization is maintained by the MPs within the overlay network allowing the above network layers and actual application to be completely isolated form the task. As a result, applications are not even aware of the SSON or Ambient Network; a part of the essence of network-side media processing.

2.4 Summary and Discussion

This chapter has introduced and presented some of the significant challenges of networked media, which range from mobility management, delivery standards, QoS, to media adaptation. First discussed was the definition of networked media and the significant challenges imposed by user mobility. Defined was the concept of Universal Mobility and how it can be managed at the network (i.e. MIP) and application (i.e. SIP) layers of the TCP/IP protocol stack. MIP and SIP were evaluated and compared and

concluded was that optimal mobility management shall arise out of multilayer management approaches. Second multimedia delivery standards were introduced, pointing out where improvements could be made. Specifically, SIP, IMS, and CDNs were discussed, which lead to the introduction and definition of a QoS model for media delivery. Concluded in this discussion was the complexity of such a model, requiring innovations in cross-layer approaches promoting layer interactions and global optimization techniques within TCP-IP/UDP-IP. Last, the SMART framework was introduced and defined as part of the Ambient Networks Integrated Project, in confronting many of the issues highlighted in this chapter. Provided by SMART using overlay network technology is the media processing capabilities necessary in media delivery. Concluded in this section was the significant need and importance of network-side processing functionality.

A main theme in this chapter has been the need to cater and adapt media to the user and their environment, with an emphasis on mobility. SMART points out that media shall always be routed to the most suitable end devices, and that media will be adapted to the capabilities of respective end devices. However, how such devices and respective capabilities are discovered and composed is not addressed. Nor is the need for the intelligent and automatic composition and management of devices and the services they provided (e.g. audio/video output services) to create the best possible service configurations at the time of need, with limited to zero human intervention. In the remainder of this dissertation, we extend this ideal and provide a true and complete scenario for the future of networked media consumption.

Chapter 3

State of the Art and Related Work

In this chapter we define the virtual device problem and provide an investigation into existing research. In addressing the technical aspects of the problem, we discuss service oriented pervasive computing with a specific emphasis on service oriented architectures and large projects based on its principles. Moreover, discussed are the emergence of networked appliances and networking technologies, setting a foundation for new research in the virtual device area. Being a broad area, we refine the virtual device problem into the sub-problem of service discovery and composition, addressing each of these issues in both infrastructure and infrastructure-less environments. Furthermore this chapter serves to define the requirements of virtual device management pointing out the difficulties current approaches present in satisfying these requirements, whereby our dissertation attempts to resolve such difficulties.

3.1 Virtual Device Origins: A Technical Perspective

The vision of *pervasive computing* implies environments populated with computing and communication facilities gracefully integrated with human users [47]. The convergence of powerful, small, affordable computing devices with network capabilities and software, seamlessly adapting them to the surrounding environments, shall facilitate this vision. Such devices may range from resource rich devices such as a computer workstation or

laptop, to increasingly constrained devices such as smart phones or mobile media devices. Constraints could appear in the form of processing power, capacity, load, display capabilities, battery power, and bandwidth, for example. Moreover, devices may be stationary (e.g large screen displays or surround sound unit) or mobile (e.g. personal media player). However, all such devices shall provide hardware and/or software functionalities to the environment through the connection of heterogeneous networks (e.g. WAN, LAN, PAN, and etc.). A user carrying a pervasive device moves from environment to environment building pervasive applications realizing user tasks in the form of virtual devices combining the functionalities of pervasive devices and adapting the resulting combination to the specifics of each environment.

The main challenges of such a vision include:

- *Environment heterogeneity*: The fact that pervasive devices and their functionalities are heterogeneous in terms of underlying technologies restricts their integration in realizing user tasks.
- *Environment dynamics*: As new devices may appear in the network and other devices disappear due to lack of resources or transmission range, for example, the environment is quite dynamic, perceived in terms of the number and lifetime of pervasive functionalities a user can access at a specific time and location.
- *Device resource constraints*: With high amounts of mobility, the probability of the realization of user tasks involving devices with limited resources is high and must be considered.
- *User centrism*: Ensuring that selected functionalities best conform to the user's needs is of highest priority, as in pervasive environments the user is the center of attention. It is imperative that the user be served as seamlessly and as naturally as possible, with efficient solutions involving minimal user intervention and acceptable response times.

Pervasive functionalities involving hardware and/or software functionalities can be abstracted as *services* catering to the service-oriented pervasive computing paradigm.

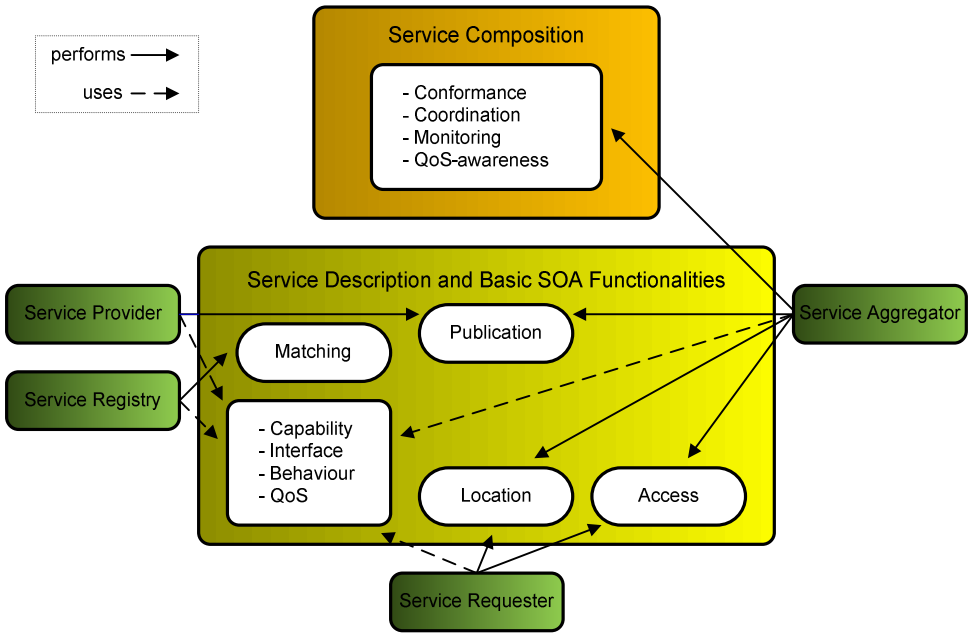


Fig. 3.1. SOA conceptual elements (redrawn from [48]).

3.1.1 Service-Oriented Pervasive Computing

Pervasive environments can be modeled using the Service-Oriented Architecture (SOA) [6]. SOA abstracts pervasive functionalities as services, whereby services are independent entities with well defined interfaces able to be accessed without any knowledge of their underlying technologies.

SOA is composed of four basic actors:

- *Service Provider:* The role assumed by an entity offering a service.
- *Service Requester:* The role assumed by a client wishing to consume a service.
- *Service Registry:* The role assumed by an entity maintaining information on available services and means of access.
- *Service Aggregator:* The role assumed by an entity which composes existing services in offering a new service to the service requester.

In SOA, services are structured utilizing a service description formalism or language specifying service capabilities, interface, behavior, QoS, addressing, and etc. From a conceptual perspective, SOA can be viewed as shown in Figure 3.1, whereby each actor is associated with the various functionalities defined bellow.

- *Service Publication:* the process of registering a service in a service registry.
- *Service Location:* the process of retrieving desired services from a service registry
- *Service Matching:* the process of selecting the best conforming service with respect to a service request.
- *Service Access:* the process of establishing a connection with a selected service.
- *Service Composition:* the process of integrating multiple services in a single composite service, which requires four sub-functionalities.
    - *Service Conformance:* the process of ensuring the integrity of the composite service.
    - *Service Coordination:* the process of coordinating service executions taking part in the composition.
    - *Service Monitoring:* the process of observing the execution of a composite service in determining the possible need for adaptation.
    - *QoS-awareness:* the process of ensuring that the composite service fulfills the specified QoS requirements.

From a pervasive computing perspective, the SOA model can be abstracted as shown in Table 3.1. A pervasive device is regarded as a *service provider* providing a pervasive functionality in the form of *service capabilities*. Services are composed to form a pervasive application realizing a user task, where the user is the *service requester*. Pervasive functionalities are advertised through *service publication*, such that *service location* plays a large role in identifying relevant pervasive functionalities. User task

Pervasive Computing	**SOA**
Pervasive Device	*Service Provider*
Pervasive Functionality	*Service Capability*
Pervasive Application Realizing a User Task	*Composite Service*
User	*Service Requester*
Advertising Pervasive Functionalities	*Service Publication*
Identifying Relevant pervasive Functionalities	*Service Location*
User Task Realization	*Dynamic Service Composition*

Table 3.1. SOA model abstraction [48].

realization is ultimately achieved though *dynamic service composition*.

In the following section we analyze a series of prominent projects adopting the Service Oriented Architecture.

3.1.2 SOA-based Prominent Projects

In this Section, we overview the Aura, Gaia, PICO, WSAMI, and Oxygen projects and provide a discussion.

3.1.2.1 Aura

The Aura project [7] argues human attention as the most precious resource in computing, and that resource rich environments made plentiful through Moore's law should be exploited for reduced load on human attention. The project addresses this issue by modeling daily user tasks as abstract and transparent software applications. The user is able to dynamically realize these tasks without the need for configuration and reconfiguration of the applications such that Aura performs automatic configuration and reconfiguration of pervasive computing environments according to the user's task and intent. Tasks are composed of services available in the environment, whereby tasks are capable of adapting themselves to the resources available in the environment. Moreover, Aura provides the ability to renegotiate task support based on variations of service capabilities and resources. In making composition decisions, Aura utilizes preference information with regards to service and service interconnection, components to supply required services, and acceptable QoS levels and tradeoffs. Furthermore, Aura is capable of capturing the user-level state of a task as the user moves from one environment to another, switches between tasks, or upon reconfiguration of a task. Aura takes a centralized approach, such that a number of central entities concentrate knowledge and coordination of the functionality of the environment.

3.1.2.2 Gaia

The Gaia project [8] abstracts physical spaces into interactive systems, termed *active spaces*, operating as a complex single high level device. Coordinated by Gaia are the devices and software components contained in the active space, allowing for the dynamic deployment and execution of software applications. This involves user assisted or automatic mapping of an application to available resources in the active space. Moreover, Gaia supports the reconfiguration of existing composed applications as well as their mobility between active spaces. Gaia enables the development, deployment, and execution of distributed, multi-device, mobile-applications within active spaces. Gaia also takes a centralized approach, such that incoming devices are sensed by a presence manager and tracked using a heartbeat mechanism. Furthermore, Gaia supports application mobility between active spaces through application suspension and resumption.

3.1.2.3 Pico

The Pervasive Information Communication Organization (PICO) project [9] focuses on time-critical applications in creating mission-oriented dynamic communities of autonomous software entities through the use of two high level components: *delegents* (intelligent agents) and *camileuns* (connected, adaptive, mobile, intelligent, learned, efficient, ubiquitous nodes). PICO application domains include telemedicine, military, and crisis management. Through the dynamic creation and composition of delegent communities PICO is able to perform tasks on behalf of the user and handle issues such as the adaptation to dynamically changing pervasive environments.

3.1.2.4 WSAMI

The WSAMI project [10] is a web services based technique for the abstract specification of pervasive computing applications in the form of software architectures and their dynamic composition based on environment availabilities. Through systematic

customization WSAMI enforces quality of service for deployed applications in terms of security and performance. Unlike the other discussed projects, WSAMI is deployable in infrastructure-less environments including resource-constrained devices for the formation of decentralized pervasive computing environments. More specifically, WSAMI is based on the *group application model*, which enables the structuring of group applications executed by ad hoc dynamic groups of peer-to-peer interacting wireless devices. Dynamic behavior is based on the parameterized specification of the group membership property, which is enriched to include QoS, security, and performance information.

3.1.2.5 Oxygen

The Oxygen project [11] focuses on enabling pervasive and human-centred computing. Specially developed are computational devices called Enviro21s and *Handy21s*, whereby *Enviro21s* are embedded in the environment and Handy21s are carried by the user allowing them to interact with the environment and perform tasks. All aspects of the Oxygen project rely on networking and software technologies developed specifically for the project creating extremely rich pervasive environments, which further include self-configuring networks and self-adaptive software. Developed as part of the project are advanced, highly adaptive, user centric applications, relying completely on developed technologies with specific interfaces communicating over specifically developed networks.

3.1.2.6 Discussion

The above projects touch on various aspects of the pervasive computing vision including mobility, ubiquity, heterogeneity, ad hoc nature, user centrism, and dynamics. In terms of *mobility*, the Aura and Gaia project specifically introduce the notion of application mobility, whereby applications are associated to users, and as users move between environments, composed applications are saved, suspended, and resumed. However, current methods are not elaborated and rely on centralized and commonly accessible file

systems. Similar techniques exist in the PICO and Oxygen projects. WSAMI integrates user and device to achieve user and device mobility, whereby the infrastructure itself is also mobile (i.e. assembled from mobile devices). In terms of *Ubiquity*, Aura, Gaia, PICO, and Oxygen projects deploy rich functionality into multiple centralized environments interconnected through file sharing. This attribute limits the range of deployment, whereby projects such as WSAMI rely on the universality of the Internet to provide ubiquitous environments. However, WSAMI trades range of deployment for less rich environments, as compared to the other discussed projects. The issue of *heterogeneity* is not handled by any of the discussed projects as they all impose homogeneous system infrastructures, although WSAMI relies on standardized web services, making the solution slightly more pervasive. In terms of achieving *ad hoc nature*, all discussed projects employ service discovery and application mapping. However, heterogeneity issues exist with regards to the solutions being restricted to compositions limited to functionally and syntactically compatible service components. *User centrism* is present in terms of a system's reactive or proactive behavior with regards to interpreting user preferences, context, and intent, in order to configure computing environments to support user tasks. However, proactively exploiting a user's context and intent remains a challenge. Lastly, *dynamics* are well addressed with the dynamic configuration and reconfiguration of applications and system services.

3.1.3 The Emergence of Networked Appliances

A network appliance is defined as a dedicated function consumer device with an embedded processor, a network connection, and the ability to disperse its capabilities within the network, allowing other devices to combine and use them to solve some given task. In this manner, a networked appliance's functionality is distributed, enabling it to be controlled, monitored, managed, and extended beyond what it was initially designed to do.

Networked appliances are considered to be one of the next major Internet growth areas, with various factors stimulating there emergence [2]. For one, the cost of network-capable electronics is steadily decreasing, with better performing, highly reliable, and

lower power consuming microcontrollers with integrated network interfaces becoming the norm for a wide variety of network technologies. Moreover, the realization of continuously available network connectivity coupled with an increase of online data availability is causing a consumer demand for the seamless integration of networked services into their lifestyle. Furthermore, mass market opportunities agreeable to appliance technology are presenting themselves, such as home power consumption control, pay-per-use technologies, service model sales for major domestic appliances, and digital rights management.

In the pursuit of home networking solutions, a plethora of wired and wireless infrastructure and network protocols have been introduced, some of which are shown in Table 3.2. From this Table one can notice that there is a vast amount of heterogeneity in terms of transmission medium, speed, and max distance; each of which is associated with various constraints such as cost, performance, security, and power consumption.

Technology	**Transmission Medium**	**Transmission speed**	**Max distance**
Bluetooth	Radio frequency	1Mbit/s–10Mbit/s	10m–100m
Ethernet	Unshielded twisted pair	10Mbit/s–1Gbit/s	100m
	Optical Fiber	1Gbit/s–10Gbit/s	2km–15km
HomePNA	Telephone line	10Mbit/s	300m
IEEE 1394 (FireWire)	Unshielded twisted pair / Optical fiber	400Mbit/s–3.2Gbit/s	4.5m–70m
IEEE 802.11 (WiFi)	Radio frequency	11Mbit/s–248Mbit/s	30m–100m
IrDA	Infrared	9600bit/s–4 Mbit/s	2m
LonWorks	Twisted pair / Electrical wiring / Radio frequency / Coaxial	1.70kbit/s–1.28Mbit/s	1500m–2700m
USB	Twisted pair	12Mbit/s–480Mbit/s	5m
X10	Electrical wiring	50bit/s–60bit/s	80m
Zigbee	Radio frequency	20kbit/s–250kbit/s	10m–75m

Table 3.2. Wired and wireless infrastructure and network protocols.

The emergence of networked appliances coupled with large amounts of heterogeneous wireless infrastructure is generating ideal conditions for achieving the vision of the truly pervasive virtual device.

In the remainder of this chapter we take a deeper look into one of the main aspects of the virtual device problem, mainly allowing for the automatic composition of devices, and the services they provide, to create dynamic service configurations; otherwise known as service discovery and composition.

3.2 Service Discovery and Composition

Service discovery and composition is a significant and relevant area of research [49-55], studied extensively in the context of web services. Research in this area has followed two paths, mainly service and composite service description and matching, and service discovery and composition architectures, with the majority of the research being conducted for infrastructure-based environments. Although research for infrastructure-less environments exists, it is still a relatively new topic. We begin this section by examining service discovery and composition in infrastructure-based environments.

3.2.1 Infrastructure-based Environments

With regards to service discovery and composition, prominent service description languages exist for describing web services in a dynamic manner, and include the Web Services Development Language (WSDL) [56], and the DARPA Agent Markup Language for Services (DAML-S) [57]. WSDL is an XML-based language describing network services as a collection of endpoints utilizing document and procedure–oriented messages to operate. DAML-S is a project focusing on the standardization of the Web Ontology Language (OWL) [58] in describing information available on any data source for exchange and understanding between devices, without human intervention. OWL can be defined as a set of XML elements and attributes, which through standardized meanings are used to define terms and their relationships. More specifically, with regards to composition, languages to formally specify services and composite services

exist, including the Web Services Flow Language (WSFL) [59], an XML-based language for the description of web services compositions, and the Business Process Execution Language for Web Services (BPEL4WS) [60], a language for the formal specification of business processes and business interaction protocols, facilitating the expansion of automated process integration. Complex planning engines have also been developed utilizing service descriptions to generate declarative specifications of workflows for the composition of services [61, 62].

In terms of service discovery and composition architectures, the majority of solutions are designed for wired infrastructures, utilizing concepts such as central lookup servers for service registration, discovery, and composition, assuming stable nodes connected by reliable communication channels. These include Jini [12], Salutation and Salutation-lite [13], UPnP [14], and the Service Location Protocol [15], among others [16-17]. Such techniques often involve preconfigured composition managers residing on high resource (e.g. memory, bandwidth, processing power) dedicated machines performing service coordination and management. Moreover, these approaches do not cater to highly pervasive, mobile, and ad-hoc environments, leaving themselves vulnerable to issues such as central points of failure, mobility, and fault management.

3.2.2 Infrastructure-less Environments

As previously mentioned, service discovery and composition in infrastructure-less environments is a relatively new area of research. Still in early phases, existing work [63-64] rely too strongly on broadcast-based techniques, prevalent in ad hoc networks, which lead to scalability and efficiency issues. However, the work of Chakraborty *et. al*. [65] makes serious contributions towards this end, with a group-based service discovery protocol based on the concepts of peer-to-peer caching of service advertisements and group-based selective forwarding of service discovery paths. Chakraborty *et. al*.'s protocol incorporates the principals of reactive systems, and consists of mobile nodes, representing various devices in the environment, providing single or multiple services, able to be invoked by peer nodes. These nodes are connected together using ad-hoc network protocols, and the discovery and integration of services allows for composite

services. Lacking in this this work, however, is the discovery and composition of services with regards to specific user and task needs and situations.

In terms of standardized protocols such as the Bluetooth Service Discovery protocol [66] being extended into ad hoc environments, reliance is also heavy on broadcast, and moreover, only basic identifier matching is supported. More recent works, including the MobiLife Integrated Project [67], Intel Research's Dynamic Composable Computing [68] project, and Namman *et. al.*'s Flexible Service Composition Framework [69] are quite promising, such that all three of these works offer contributions towards decentralized ad hoc compositions tailored to specific user needs and situations.

The MobiLife Integrated Project brought advances in mobile applications and services within reach of users in their everyday lives by innovating and deploying new applications and services based on the evolving capabilities of 3G systems and beyond [67]. The Project captured aspects of the virtual device problem with regards to its contributions in the area of multimodality and personalization. Developed were theoretical frameworks for satisfying the mobile user experience. Strong emphasis was placed on making the best use of available environment devices, while ensuring that applications offered to the user were tailored to their specific needs and situations. Separating MobiLife from similar projects is the inclusion of mobile devices as core components, and the consideration for device mobility and the need for continuous adaptation. Unfortunately, much of this work remained in the theoretical realm.

Dynamic Composable Computing (DCC) is a term defined by Intel Research and refers to the impromptu assembly of a logical computer from the best set of wireless parts available nearby [68]. In their work, Intel Research builds on existing standards to develop an architecture and prototype that enables users to quickly connect mobile devices together. In particular, Intel Research develops a layer-2 service discovery scheme [68], which integrates IP service advertisement with the layer-2 advertisement and discovery process. This solution serves to reduce the inefficiency of broadcast discovery techniques in ad hoc networks. Unfortunately, Intel Research still relies on manual and named compositions requiring heavy user involvement, although future work is in place to investigate ranking-based compositions.

Namman *et. al.*'s Flexible Service Composition Framework [69] proposes a solution which incorporates a hybrid (i.e. centralized and distributed) approach to service composition. The approach utilizes as default a centralized technique to enable automatic service compositions of networked appliances in heterogeneous environments. Should the central server become unavailable, local information is used to perform the composition process in combination with an interrogation procedure between neighboring nodes. Although Namman *et. al.'s* solution provides adequate support in an environment with limited central server interruption, the technique would inevitably suffer in an environment too dynamic for a centralized server to exist at all.

An interesting research question is the potential use of overlay networks in the composition of virtual devices. Overlay networks can be used to enable the flexible configuration of virtual networks composed of Overlay Nodes on top of underlying physical networks, and transparently provide data processing capabilities such as media transformation, splitting, and routing. Such media processing is necessary and can be exploited in the virtual device problem. In the following Section, we investigate the overlay network concept.

3.3 Service Overlay Networks

The term overlay network can be defined as a virtual network of nodes and logical links constructed over an existing network, for the purpose of implementing a network service not available in the existing network. Moreover, an overlay network can be viewed as a application layer Internet separating the physical layer from applications and supporting customization in meeting and optimizing specific functionalities. Overlay networks can be deployed without the need for ISP cooperation, nor the deployment of new equipment or even software and protocol modifications [70-72]. Such frameworks can be classified into application specific overlay networks and generic overlay networks.

Application specific overlay networks are designed and tailored to specific applications, such as multicasting [73-74], content distribution networks [21-22], and peer-to-peer file sharing [75]. *Generic overlay networks* [76-82] are designed to support a variety of diverse applications, where knowledge is shared utilizing an intermediate

layer measuring a number of network properties. Although a generic overlay network efficiently provides shared knowledge across various applications, it lacks the ability to support the specific demands for individual services, which is supported in application specific overlay networks. However, in a tradeoff, application specific overlay networks lack generality and are less efficient.

The benefits of overlays come at the cost of increased overhead and complexity. If improperly managed, overlay networks can lead to network overload, bottlenecks, and redundancy at the IP layers. This makes *overlay management* a necessary incorporated mechanism. Management techniques must account for the four phases overlays encounter in their lifetime: creation, optimization, adaptation, and termination. *Creation* involves the generation of routing tables in each node participating in the overlay along the end-to-end path. *Optimization* is the establishment of this end-to-end path such that particular QoS metrics are optimized. *Adaptation* enables the overlay to modify its behavior, such that it reflects a change in the overlay's environment. *Termination* involves the release of claimed resources and the updating of routing tables.

Well studied and existing techniques for overlay management mainly fall within three categories: policy-based, active network technology-based, and automated. *Policy-based* approaches offer flexible and customizable management solutions allowing for the on-the-fly configuration of network entities [83-84]. Widely supported by standard organizations such as the IETF and the DMTF, policies allow administrators to define sets of rules enabling the control of network entity behaviors. These defined rules are translated into component-specific polices stored, retrieved, and enforced as needed. *Active Networks technology-based* approaches incorporate frameworks where network elements (e.g. routers and switches) are programmable and programs can be injected into the network achieving higher flexibility and new capabilities [85-89]. All active nodes are equipped with Node Operating Systems and one or more Execution Environments (EE), where EEs define distinct virtual machines (i.e. programming interfaces) for Active Applications to be developed on, providing particular end-to-end services. *Automated* management approaches go beyond simple adaptation algorithms and towards the achievement of self-adaptation [90-97]. With a focus on QoS, such approaches attempt to control and manage the creation, customization, and support of

services for demanding applications. However, scalability is still an important issue in such management techniques.

In terms of management we focus on the *autonomic* management of overlay networks. *Autonomic Computing* (AC) was developed by IBM in 2001 [98], whereby they define an AC system as a system that is aware of itself and the environment, configuring and reconfiguring itself under varying and unpredictable conditions, with the ability to protect and heal itself while keeping complexity hidden. Since the appearance of the concept, a variety of works have attempted to incorporate AC. R. Farha *et. al.* provide a generic architecture for autonomic service delivery, defining a resource management model based on virtualization [99]. E. Kaston *et. al.* propose pattern classification and clustering techniques supporting online decision making and incremental learning in autonomic system [100]. Bahati *et. al.* propose a autonomic approach to configuring autonomic elements in Apache web servers in enforcing required behaviors [101]. Pena *et. al.* define UML-based models in specifying autonomic properties in the modeling, specifying, and deploying of policies [102]. Other projects including the AC concept include Service Clouds [103], Autonomia [104], GridKit [105], Auto-Mate [106], and Unity [107]. A detailed survey of AC is provided in [108].

3.4 Summary

This Chapter has presented the virtual device problem from a technical perspective. In order to make this vision a reality, strong suites of complex systems must be made to operate, cooperate, and communicate without user intervention. Recent emergence of networked appliances and networking technologies as well as advances in service oriented architectures is a driving force to this end. However, much work is to be done in achieving truly pervasive computing experiences; accounting for user dynamics, mobility, and intent in infrastructure-less environments.

Chapter 4

Autonomous Management of Virtual Devices

Personal computing is experiencing a shift from conventional computer-centric approaches, where the overall user experience and quality of service is determined by the capability of a single device, towards human centric approaches forming implicit devices, also known as virtual devices or virtual appliances. Virtual devices exploit the strengths of a series of strong specific networked appliances for enhanced user experience and quality of service. In this chapter, we contribute towards the autonomous management of virtual devices and set a foundation for the remainder of the thesis, moving away from infrastructure based schemes with heavy user involvement to decentralized and zero touch (i.e. no user involvement) solutions. In doing so, we present the components and methodology behind a policy-driven autonomous framework for the dynamic discovery, selection, and composition of multimodal multi-device services. The framework operates in an ad hoc network setting and introduces a Service Overlay Network (SON) based definition of a virtual device.

4.1 Introduction

The continued proliferation of *networked appliances*, or dedicated function consumer devices embedded with processors and network connection capabilities is facilitating the realization of multimodal multi-device environments allowing user interfaces to react to the changes of available devices and modalities within the immediate environment of the user. This is a concept referred to as the *virtual device*, whereby a virtual device provides a dynamic system of strong specific networked devices, both fixed and mobile, offering distributed multimodal capabilities, able to provide a coherent and surrounding interface to the user [23]. The creation, maintenance, and termination of a virtual device is derived from contexts regarding application tasks, service constraints, and user preferences.

Various aspects of the virtual device problem are visible within the home networking research area [20]. The focus of these works range from advanced human-computer interaction, to networking and automatic interoperation of devices. However, these works tend to be orthogonal to the development of enabling platforms for interoperation, and only manage to provide basic interoperability through centralized approaches, not providing support for ad hoc environments. The iRoom [109] and Pebbles [110] projects, for example, both focus on linking various computing elements into an interactive workspace, allowing multiple participants to interact in a shared workspace environment. However, this is achieved using a centralized infrastructure, not considering characteristics of flexible composition including device discovery and composition. Easyliving [111], a Microsoft project for the dynamic aggregation of diverse I/O devices into a coherent user experience, does focus on various vision and interference based techniques for sensing users, enabling the dynamic coordination of devices in a home environment; but, similar to the two previous works, also achieves the preceding using a fixed infrastructure for composition.

More recent works, including the MobiLife Integrated Project [67], Intel Research's Dynamic Composable Computing [68] project, and Namman *et. al.*'s Flexible Service Composition Framework [69] are quite promising, such that all three of these works offer contributions towards decentralized ad hoc compositions tailored to specific user

needs and situations. Moreover, these works include mobile devices as core components. However, yet to be considered is the derivation of an autonomous approach to virtual device management.

In this chapter, we address a broad view of the virtual device problem and develop a framework for virtual device management. In finding more reactive and decentralized solutions to the problem, we focus on distributed virtual device composition in mobile ad hoc network environments (MANETs). Moreover, unlike existing works targeting service composition in MANETs (e.g. [65,112]), we focus on finding the 'best composition possible' as opposed to simply 'a composition that fits'.

In approaching virtual device management from an autonomous perspective, we derive a policy-based composition framework we call the Autonomous Virtual Device Management Architecture (Auto-VDMA) for the dynamic discovery, selection, and composition of multimodal multi-device services present in a user's changing environment. In Auto-VDMA, the fusing of multimodal multi-device services lies in the self-configuration of *service overlay networks* (SONs) [113]. SONs enable the flexible configuration of virtual networks composed of Overlay Nodes (ONodes) on top of underlying physical networks, and transparently provide network-side data processing capabilities, performing value-added processing, such as media transformation, splitting, and routing. We therefore define, in the context of our framework, a virtual device as a dynamic policy driven SON providing an end-to-end media delivery path to form a multimodal multi-device environment. Using this definition, we build a framework composed of a hierarchical structure of distributed elements, including autonomic elements, all working towards the self-management of virtual devices.

The resulting contribution of this chapter is a novel SON-based definition of a virtual device applied to an autonomous policy driven framework for virtual device management.

The rest of this chapter is organized as follows. The following Section presents a scenario and system requirements. Section 4.3 describes our approach and framework towards autonomous virtual device management, and Section 4.4 provides a summary.

4.2 Scenario and Requirements

We present a scenario to visually clarify the concept we are trying to achieve, followed by the defining of main system requirements.

4.2.1 Scene One – Home

Mr. Smith, carrying an Auto-VDMA enabled mobile companion device, similar in shape and form to today's smart phones, enters his home. Having just returned from the grocery store, Mr. Smith hastily puts away his perishables. At that moment, a video call comes into his companion device. Mr. Smith answers the call in virtual device mode. Immediately, the caller's video is displayed on the LCD embedded in Mr. Smith's fridge, and the caller's voice is projected out of the speakers mounted in the kitchen as part of Mr. Smith's home audio solution. Mr. Smith proceeds to greet the caller as his voice and picture are picked up by his embedded home monitoring system. The call is brief, and Mr. Smith receives an invitation to a new restaurant in New York, a city which he is flying to that evening.

4.2.2 Scene Two – Car

Now on his way to the airport, Mr. Smith enters his vehicle. Using his companion device, he launches the music application in virtual device mode. His companion device begins to play his favorite music files; not on the device itself, but using the stereo system of the vehicle. As Mr. Smith is driving and listening to his music, a call comes in, but before he is even aware of the call, his companion device has already lowered the volume of the music, redirected the voice output of the call to the two speakers closest to Mr. Smith, and activated a microphone embedded in the steering wheel of the vehicle for the redirection of voice input. Mr. Smith answers the call, and a hands-free voice communication session is established.

4.2.3 Scene Three – Airport

Currently at the airport, waiting for his flight in an executive lounge, Mr. Smith, knowing little about the restaurant he is to dine at this evening, decides to visit the restaurant's web site. Entering the lounge's media room, Mr. Smith picks up his companion device and activates the Internet browser application in virtual device mode. Immediately, a flat panel television displays his home page, a greeting message is heard out of a theatre surround sound unit, and a keyboard and mouse embedded in a coffee table light up. Mr. Smith proceeds to view the restaurant's web site.

4.2.4 Requirements

The UML use case diagram in Figure 4.1 depicts the six main functional requirements of the system. The diagramed use cases involve three actors, namely mobile user, mobile device, and location. A mobile device is carried by a mobile user moving from one location to another.

4.2.4.1 Functional Requirements

- **Use Case #1:** *The system reacts to the user's movement from one location to another as implicit input, such that the behavior of the mobile device is automatically adapted to meet the availabilities of the user's current location (i.e. implicit output).* Use case #1 represents the interaction between the user and the system.
- **Use Case #2:** *The user interacts with the mobile device activating applications without any further involvement in the process of composing a virtual device.* Use case #2 represents the zero touch aspect of the system, such that the remaining use cases are completely transparent to the user, whereby the user maintains a relative one-to-one type relationship between the computer and themselves.
- **Use Case #3:** *The mobile device creates a virtual device within the environment for the user to execute their desired task.* Use case #3 signifies the ability of system to

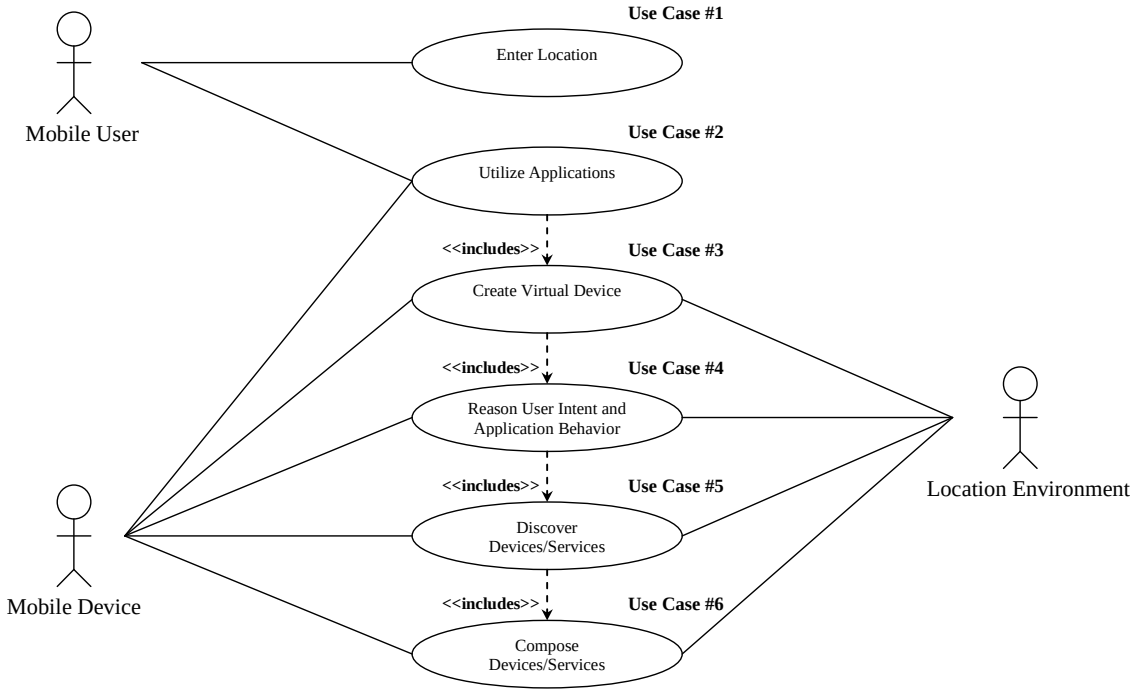


Fig. 4.1 System use case diagram.

compose a virtual device from the devices in the environment and the services they provide.

- **Use Case #4:** *The mobile device reasons appropriate virtual device behavior, based on user intent and application requirements.* Use case #4 implies the ability of the system to intelligently interpret user and application context and reason situational behavior.
- **Use Case #5:** *The system discovers all devices and services necessary in creating a virtual device satisfying user intent and application requirements.* Use case #5 signifies the service discovery aspect of the system, whereby discovered, are services necessary in composing the best possible virtual device at the time of need.
- **Use Case #6:** *The system composes discovered devices and services necessary in creating a virtual device satisfying user intent and application requirements.* Use case #6 signifies the service composition aspect of the system, whereby composition involves rectifying capability differences (e.g. supported codecs, resolutions, bandwidth requirements, etc.) and accounting for the splitting or joining of media for proper consumption.

4.2.4.2 Non-functional requirements

In achieving the use cases depicted in Figure 4.1, the following non-functional requirements are imposed:

- **Light weight:** No physical sensors or heavy computational resources shall be required by the mobile device. Distributed resources in the location environment shall be exploited to achieve desired goals.
- **Flexible:** Multiple virtual device compositions shall satisfy user intent and application requirements, whereby each achieves this end with varying degrees of QoS.
- **Automatic:** A user shall have a minimal role in the process of virtual device creation, maintenance, and termination. The role of a user shall only extend to the defining of preferences and the response to solicitation for confirmation.
- **Seamless:** From a user perspective, utilizing an application in virtual device mode should be as seamless as portrayed when utilizing an application on a single device.

4.3 Autonomous Virtual Device Management

In addressing virtual device management, we consider an environment made up of mobile nodes representing various devices providing single or multiple services, able to be invoked by peer nodes. Such nodes are joined together using ad hoc network protocols, and a virtual device is provided to the user through the composition of their services, satisfying an atomic task. We define an *atomic task* as an application level generated task, independent from any other task, requiring one or more atomic services. We define an *atomic service* as a service residing on a single node, which can consist of further components if and only if they too reside on the same single node.

An Atomic task is represented in the form of a *task composition specification (TCspec)*. A TCspec is a specification of service and service property requirements, characterized as policies divided into three categories: task requirement, service constraint, and user preference. *Task requirements* represent the minimum services required for task satisfaction (e.g video input/output service). *Service constraints*

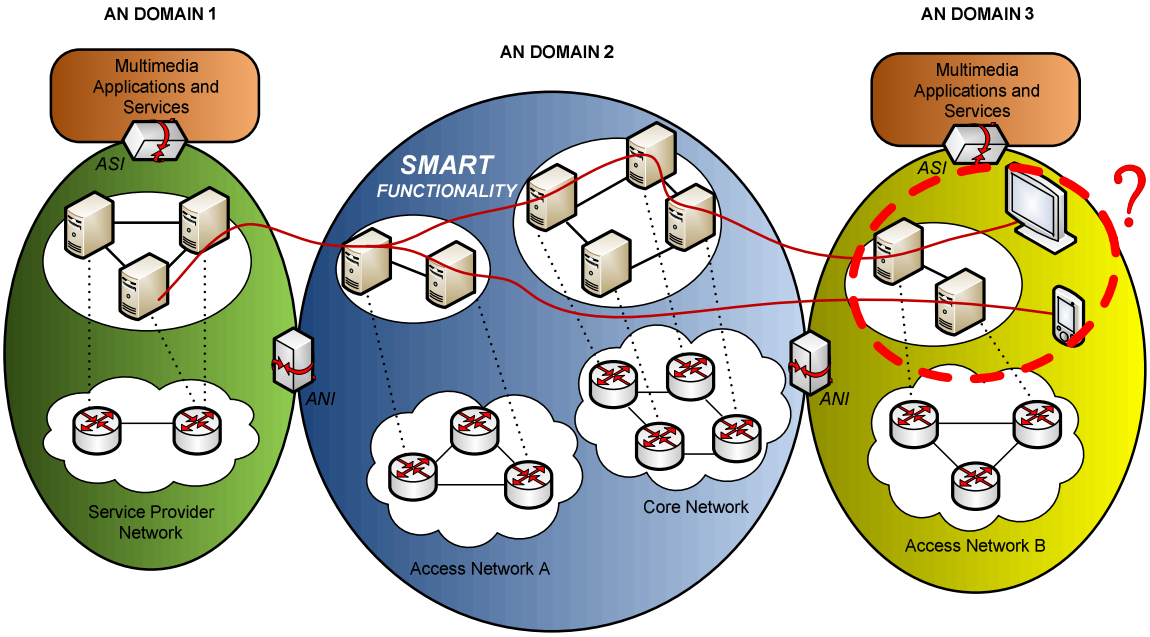


Fig. 4.2 Incomplete SMART vision.

represent the necessary constraints on the services to allow the composed service to function as the application intended (e.g. “all services must reside in close proximity”). *User preferences* are service constraints controlled by the user, which are present solely to shape the user’s desired quality of Service (QoS) (e.g. video output service ≥ 1000×800 resolution). We refer to QoS, such that at any instant, numerous potential compositions subsist in the environment fulfilling a task. Separating one composition from another is the QoS coupled with the particular composition (details provided in Chapter 5).

We propose the framework shown in Figure 4.3, referred to as the Autonomous Virtual Device Management Architecture, or Auto-VDMA for short, providing a policy driven infrastructure for the autonomous management of dynamic and distributed multimodality. We model Auto-VDMA following IBM’s architectural blueprint for autonomic computing [98], formed from various manual and autonomic managers, which together provide the attributes necessary for self-management. Moreover, although designed as an independent architecture complementing various pervasive solutions, the Auto-VDMA architecture is a viable solution to be integrated into the SMART architecture in completing their vision of routing media to the most suitable end

devices, and adapting media to the capabilities of respective end devices, which is currently unaccounted for (see Figure 4.2).

4.3.1 Framework Architecture

The Auto-VDMA (Figure 4.3) architecture is a three layer framework divided into six elements, where each element is separated by respective touchpoints, exposing state and management operations used between interfacing elements.

Beginning at the lowest layer of the architecture is the *Autonomic Node Manager* (A-NM) element, encompassing the manual *Service Manager* (SM) element. Autonomic node managers are present on any and all nodes providing or consuming services. Node and service managers work together to provide a direct interface between media services and the system. The service manager exposes functions to the node manager, necessary in querying information about, requesting periodic measurements from, as well as fine tuning services. Services fall into one of four categories: Media Service Server (MSS), Media Service Port (MSP), Media Controller (MC), and Media Service Client (MSC). *Media service clients* are services that consume media (e.g. flat screen, speaker, etc.), as opposed to *media service servers*, which are services that provide media for consumption (e.g. media server). *Media service ports* are services providing media processing (e.g. media transformation, splitting, caching, routing, etc.), utilized, as media flows from media servers to clients. *Media controllers* are services that do not provide or consume media, but instead contribute towards the user control of the media (e.g. mouse and keyboard).

Above the bottom layer is the *Autonomic Composition Manager* (A-CM) element, encompassing the manual *Service Overlay Network Manager* (SONM) and forming the middle layer of the framework. Autonomic composition managers are present on nodes with adequate resources (e.g. processor, power, load) to act as brokers in the composition process, such that we take a broker-based approached to composition. Composition and SON managers interact with node managers to control the formation and existence of virtual devices by discovering the appropriate services satisfying a TCspec, and composing such services into a SON with the necessary added media

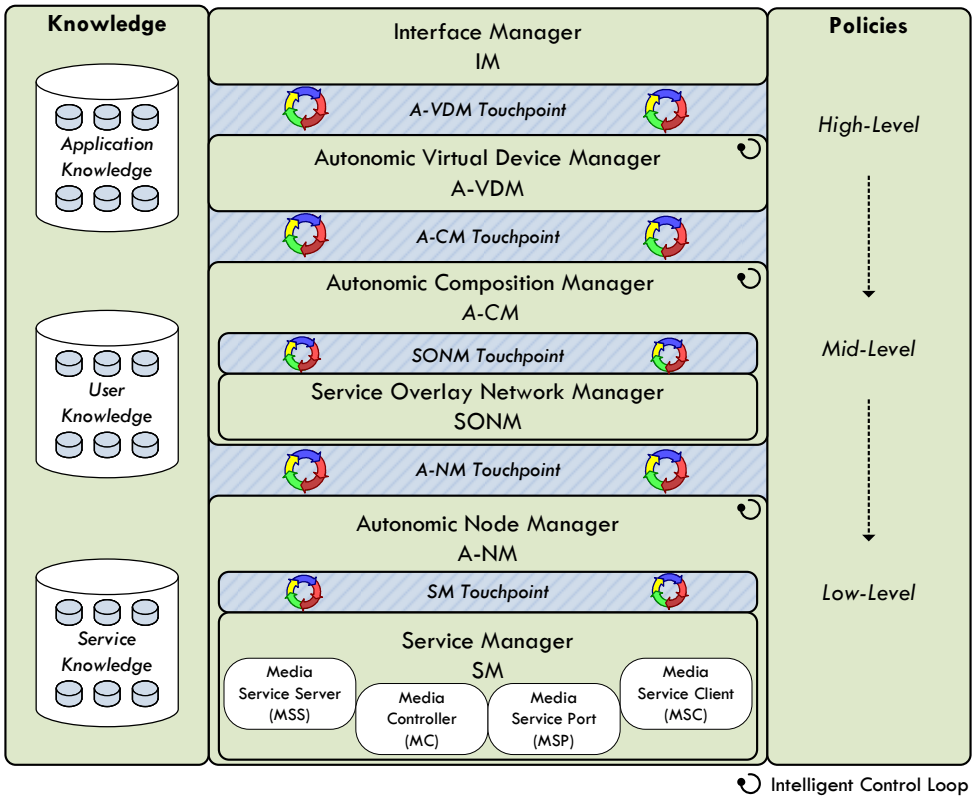


Fig. 4.3. The autonomous virtual device management architecture.

processing functionality (i.e. media service ports). Node and SON managers expose functions to the composition manager necessary in the discovery and selection of services, as well as the formation and management of SONs.

Above the middle layer is the top layer composed of the *Autonomic Virtual Device Manager* (A-VDM) element and the manual *Interface Manager* (IM) element. Autonomic virtual device and interface managers are present on nodes acting as user companion devices. The virtual device manager is responsible for promoting system-wide autonomous behavior by initiating and managing the formation of virtual devices based high level system goals provided by applications (e.g. task/service requirements) and users (e.g. preferences), obtained by the interface manager. From such information, the virtual device manager forms a TCspec, and discovers a composition manager in the network best suited to act as a broker for the particular composition. The interface manager is responsible for providing reusable virtual device functionalities to applications and users, without the need for them to explicitly implement multimodality

features themselves. Composition managers expose functions to the virtual device manager necessary in discovering a proper broker, as well as in communicating TCspecs and managing multiple virtual devices.

4.3.2 Policies and the Intelligent Control Loop

Unlike the manual managers of our framework (i.e. IM, SONM, SM) , autonomic managers (i.e. A-VDM, A-CM, A-NM) manage specific resources, whereby they collect and analyze resource details in deciding what, if any, actions need to change. Moreover, autonomic managers generate policies that reflect the required change and enforce these policies on the appropriate resources. This model of intelligence is implemented in each autonomic manager as the *intelligent control loop* (ICL). The intelligent control loop we define is shown in Figure 4.4, and is inspired from the IBM control loop [98]. ICL provides an automated method for autonomic elements to collect and analyze information, in determining if and what actions are necessary for adaptation with regards to self- management. Stimulating ICL is a distributed set of knowledge composed as policies specifying goals, objectives, profiles, and agreements. From a generic perspective, a policy can be defined as a set of constraints on the possible behaviors of a system, where system behavior is a sequence of system states [114].

Policy-based management approaches allow high-level objectives or goals to be defined without having to specify detailed system configuration information. Through a process of policy transformation [114], high-level abstractions flow through the system translated and transformed into specific forms needed by different parts of the overall system, such that enforcement modules of the lower level elements do not accept policies in the same level of detail or format that the generation module of higher level elements create. Policy transformation is responsible for bridging this gap. As policies flow and adapt through the system, a series of distributed knowledge evolves over time, used in the process of analysis, decision making, and the generation of new policies.

The creation, translation, adaptation, and deletion of policies are used by autonomic managers in relaying and orchestrating system-wide goals and actions. ICL is modeled as a group of agents with respective flows of information, as depicted in Figure 4.4. We

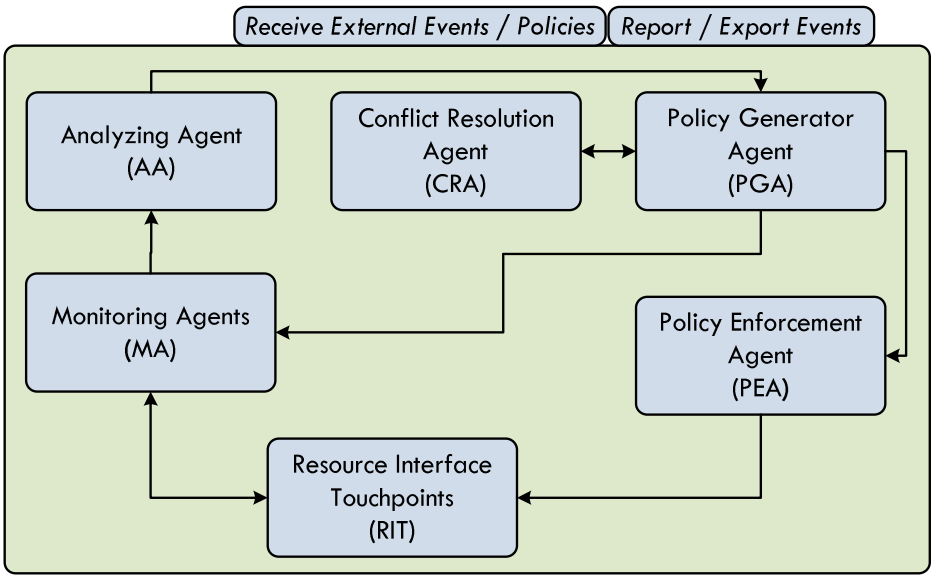


Fig. 4.4. The intelligent control loop.

briefly define each of these agents and provide an example using the autonomic node manager.

- *Monitoring Agents (MA)*: MAs are in continuous communication with the resources they are collecting information from, through respective *Resource Interface Touchpoints (RITs)*. They receive policies from the Policy Generator Agent (PGA) with respect to what data should be monitored from the resources, as well as how often this data should be reported to the Analyzing Agent (AA). Moreover, policies could represent specific thresholds dictating violations to report to the AA, based on collected data.
- *Analyzing Agent* (AA): The AA receives data from MAs and checks whether policies received from the PGA are being met. Unlike MAs, deciding whether a policy is violated or not, requires the correlation and aggregation of data with respect to various contexts. Moreover, statistical analysis may be performed. The AA reports policy violations to the PGA.
- *Policy Generator Agent* (PGA): The PGA receives high level external policies (i.e. from higher level managers), and decomposes such policies into lower level policies correctly applied to the resources it is managing. Moreover, it responds to policy violations reported by the AA. Whenever a new policy is generated, it must first pass

through the Conflict Resolution Agent (CRA) to ensure its consistency with existing enforced policies.

- *Policy Enforcement Agent* (PEA): The PEA receives CRA approved policies from the PGA that are bound for managed resources. In passing these policies to the respective RITs, each RIT translates the specified policy actions into respective resource adjustments.

The autonomic node manager's implementation of the MA module of ICL provides a method to monitor node media service resources by collecting and filtering details arriving from the service manager. Such details may include packet loss, delay jitter, throughput and etc. Secondly, the implementation of the AA module provides functions to aggregate and analyze those details based on installed policies through correlation and modeling. Based on the intelligent reasoning of the AA module and the implementation of the PGA and PEA modules, the node manager is able to derive and/or adapt specific policies, which are passed to and enforced by the service manager through the SM Touchpoint (i.e. RIT module). Furthermore, its AA module derives dynamic node profiles, identifying specific details of respective media service resources regarding attributes such as availability, reliability, security, QoS and etc. This information is used by higher layers in the process of creating and managing virtual devices. The other autonomic elements of our framework operate in a similar fashion, where A-NM and SONM are A-CM's monitored resource and A-CM is A-VM's monitored resource.

4.3.3 Phases of Composition

Virtual device composition in our framework is the product of a three phase process (illustrated in Figure 4.5) including broker arbitration, service discovery, and service integration and execution. Throughout these phases we consider the requirements of the virtual device concept, which consist of being aware of the user's continuously changing environment, immediately recognizing the expectations of the user, and instantiating a service with respect to the user's given circumstance. We structure a QoS function (defined in Chapter 5) identifying the degree of match between the requirements of a user's task (including service constraints satisfying both task requirements and user

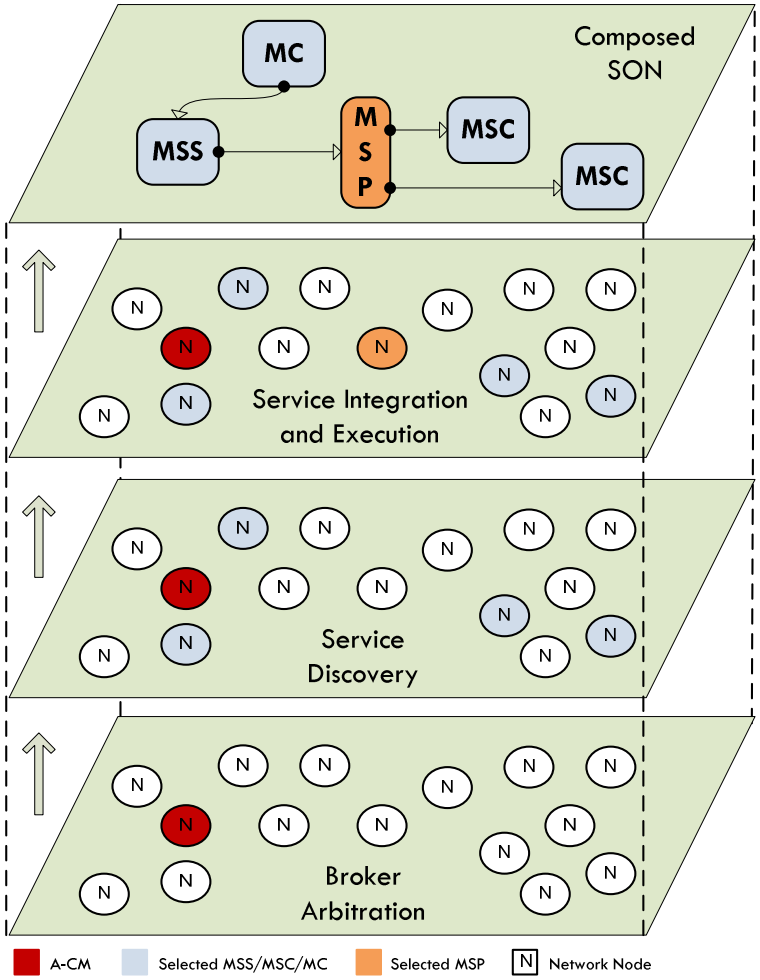


Fig. 4.5. The three phases of composition, illustrated.

preferences) and the qualities and capabilities of service composition, taking into account the current state and availability of each service taking part in the composition. We then disseminate various aspects of the QoS function into service advertisements, broker arbitration and service discovery as a means of forming the best possible virtual device at the time of need.

In *broker arbitration*, the first phase of composition, the virtual device manager elects a composition manager maximizing a potential value taking into consideration each potential broker's resources and the service providers in their vicinity. This information is calculated by each potential broker and sent to the virtual device manager. The elected broker is ideally a node with high computational power, low computation load, high battery life, and located in an area of high service density providing the

greatest opportunity for discovering services enabling the highest degree of match with regards to a user's desired task.

In *service discovery*, the second phase of composition, the composition manager discovers all necessary services (i.e. MSS, MSC, MSC) excluding MSPs, which are discovered in the following phase. The selection of services for composition is based on the formation of service composition candidates, or sets of atomic discovered services satisfying the requirements of a particular TCspec. These composition candidates represent all the possible composition permutations available in the user's vicinity, which using the QoS function mentioned above, can be ranked, therefore divulging the best available composition.

We have taken two approaches to service discovery. In our first approach, discussed in Chapter 5, discovery is based on the matching of service discovery requests with cached descriptions of services, which are updated in correlation with the group-based service discovery protocol (GSD) [115]. GSD is based on two concepts: peer-to-peer caching of service advertisements within limited vicinity and group-based selective forwarding of discovery requests. Each node manager periodically advertises a list of its services to other peer nodes within limited vicinity. Included as part of the advertisement is a list of service groups the service provider has seen in its vicinity. Grouping information is used to selectively forward discovery requests to node managers, allowing for the reduction of network-wide broadcasts and increased efficiency in discovering services.

In our second approach, discussed in Chapter 6, we examine the possibility of performing device discovery without the use of broadcast-based service advertisements by modeling and solving service discovery as a distributed constraint satisfaction problem (distCSP) [116]. By developing and applying a backtracking-based algorithm, we show the effectiveness of our method in achieving high QoS compositions without the unnecessary depletion of node resources caused by broadcast-based service advertisements.

In *service integration and execution*, the third phase of composition, the composition manager addresses the fact that discovered services (i.e. MSS, MSC, MSC) in the discovery phase, may possess capability differences (e.g. supported codecs, resolutions,

bandwidth requirements, etc.) and/or require the splitting or joining of media. This is achieved in cooperation with the SON manager, through the discovery of media service ports in the formation of a SON. We address this issue in Chapter 7, where we examine the possibility of extending our distributed constraint satisfaction problem (distCSP) model to include a method for capability reconciliation.

Ad Hoc networks are known for their highly dynamic nature. As a result, assuming that a composed virtual device will remain valid in a static form is incorrect. As the network changes, the virtual device should adapt to these changes in a manner providing minimal disruption to the user. We address this issue in Chapter 8, through the design and application of a network performance influenced QoS model for the graceful degradation and upgrade of virtual devices post-composition.

4.4 Summary

In this chapter, we have presented a framework for the autonomous management of virtual devices, incorporating a SON-based definition of a virtual device. Provided was a discussion of the autonomic elements of the framework architecture, as well as the intelligent control loop guiding their combined actions. Identified was the process of composition and how the following chapters fit into the framework, setting a foundation for the remainder of the thesis.

Chapter 5

A Distributed Protocol for Virtual Device Composition

The dynamic composition of systems of networked appliances, or virtual devices, in MANETs, enables users to generate, on-the-fly, complex strong specific systems. Current work in the development of service composition architectures in MANETs has yet to address QoS metrics for enhanced composition from a virtual device perspective. In this chapter, we present an extension to a prominent dynamic broker-based distributed service composition protocol, embedding in a distributed manner, a QoS model providing compositions that form the best possible virtual device at the time of need. Simulation results show that our protocol extension provides a high increase in QoS at a low cost in terms of increased amounts of messages and composition time.

5.1 Introduction

An open problem in achieving virtual devices is how to allow for the automatic composition of devices, and the services they provide, to create dynamic service configurations with limited human intervention. Where the aim is to automate the discovery process and enable devices to determine what services are required. In this chapter we contribute to this open issue by extending a prominent dynamic broker-based

distributed service composition protocol [65], which assumes a workflow specification of a composite service and performs the task of discovering, integrating, and executing the services. In our extension we apply into the various phases of Chakaborty et al.'s distributed service composition protocol [65], a QoS function taking into account the degree of match between the requirements of a user's task and the qualities and capabilities of service composition, including the current state and availability of each service taking part in the composition. The resulting contribution is a distributed service composition protocol tailored for virtual device composition, where the elected broker is not only resource rich and located in a high service density region, but located in a region providing the greatest opportunity for discovering services providing the highest degree of match with regards to a user's desired task. Moreover, infused into the protocol is a means of identifying the variation of similarity between two services of the same type, the current availability of each service, and how important such variation and availability is to the user, for a service composition that forms the best possible virtual device at the time of need. With this extension, the protocol remains decentralized, load-balancing, and fault tolerant. Simulation results demonstrate the effectiveness of the extension and low cost, in terms of increased amounts of messages and composition time, imposed for added functionality over the existing distributed service composition protocol.

The rest of this chapter is organized as follows. The following Section 5.2 presents related work. Section 5.3 provides a background of Chakraborty *et. al.*'s protocol [65]. Section 5.4 describes our protocol extension in detail. Section 5.5 presents our simulation results, and Section 5.6 gives a summary.

5.2 Related Work

Although works addressing device interoperability and providing interworking solutions do exist, primarily in the area of *home networking* [2], they depend heavily upon the development of enabling platforms, are centralized, and make use of overly flexible open standards, which can lead to failures between different vendors. We, as an alternative, focus on the use of mobile ad hoc network environments (MANETs), such

that there has been a very limited amount of work done in the development of distributed protocols for service composition (e.g. [65] [112] [118]), and to the best of our knowledge, none of which include QoS metrics for enhanced composition from a virtual device perspective. The main focus of most existing distributed protocols for service composition is to discover, integrate, and execute services while satisfying resource constraint issues prone to MANETs, where the quality of the composed service is out of the scope of the work. Moreover, QoS within MANETs remains a challenging problem as it requires a different approach from models assuming fixed network infrastructures. We take the work of M. Perttunen *et. al.* [119] as an example, who design a QoS model, which we feel is very well suited to the virtual device problem. However, the model is designed for, and applied to, a task-based service composition scheme for a centralized resource management technique enabling automatic service composition in smart spaces, and hence, cannot be applied to MANETs. Reason being that this model does not consider issues related to MANETs, such as mobility and varying service topologies, limited amounts of node resources (e.g. processing power/load, battery life), and single points of failure. We therefore, experiment with providing a virtual device based QoS model, able to function within MANETs.

5.3 A Distributed Service Composition Protocol

D. Chakraborty *et. al.*'s distributed service composition protocol [65] is based on the principals of reactive systems, and consists of mobile nodes, representing various devices in the environment, providing single or multiple services, able to be invoked by peer nodes. These nodes are connected together using ad-hoc network protocols, and the discovery and integration of services allows for composite services. Although nodes may differ with regards to physical properties such as computational power and battery life, there is no differentiation between a requesting source, service provider, or broker, whereby a requesting source may simultaneously be acting as a service provider and/or a broker for separate composition requests. The process of composition begins by a requesting source, wishing to compose a service, electing a broker within the network. Once elected, the broker discovers nodes providing the required services (i.e. service

providers), and integrates and executes the selected services on behalf of the requesting source.

The protocol executes through four phases. The first phase is *broker arbitration* where the requesting source elects a broker maximizing a potential value taking into consideration each potential broker's resources and the service providers in their vicinity. This information is calculated by each potential broker and sent to the requesting source. The elected broker is ideally a node with high computational power, low computation load, high battery life, and located in an area of high service density. Information with regards to service providers in a broker's vicinity is obtained by each potential broker examining their cache, which is updated in correlation with the group-based service discovery protocol (GSD) [115]. GSD is based on two concepts: peer-to-peer caching of service advertisements within limited vicinity and group-based selective forwarding of discovery requests. Each service provider periodically advertises a list of their services to other peer nodes within limited vicinity, also including as part of the advertisement a list of service groups the service provider has seen in its vicinity. The hierarchical grouping of services is exploited in the service discovery process, such that grouping information is used to selectively forward discovery requests to service providers. The overall effect is the reduction of network-wide broadcasts and increased efficiency in discovering services.

The second phase is *service discovery*, which, using GSD, discovers all necessary services. During this phase, discovery is based on the matching of service discovery requests with cached descriptions of services, where services are represented using a DAML-based semantic representation [57]. Although this nature of matching can identify two services of the same type, it is limited in identifying the variation of similarity between two services of the same type, the current availability of each service, and how important such variation and availability is to the user.

The third and fourth phases are *service integration* and *service execution*, respectively. In these phases, discovered services are selected for integration and execution. No specific selection algorithm is used, but services are selected based on various primitive cost factors, such as "nearest available service". From the selected services, an execution flow is created identifying service information, node binding,

control flow, network parameters, and other information relevant to execution. Execution occurs in a distributed manner, where the broker executes one service and transmits information received from the last executed service to the next. This sort of star-shaped execution continues as pertaining to the requirements of the execution flow. Fault tolerance is achieved through a checkpoint technique, whereby the broker periodically communicates the current state of execution, including delivery of partial results, to the requesting source. Should the requesting source loose communication with the broker, the whole process need not be restarted, but only the portion not covered by partial results. Throughout each phase, network load is controlled by regulating the number of hops involved in various aspects of broker arbitration, and service advertisement and discovery.

5.4 Distributed Virtual Device Composition

We extend the protocol to include a QoS function for task-based service composition and modify various phases of its application to the system. In this extension, which we name the Distributed Virtual Device Composition protocol (DVDC), we apply a QoS model similar to that defined by M. Perttunen *et. al.* [119] (discussed in Section 5.1). However, unlike M. Perttunen *et. al.*'s model, we relax the assumption of an infrastructure-based network environment by taking a decentralized approach allowing the model to function in MANETs. We base our system on the following entities:

- *Requesting Source* (RS): A mobile node from which a composition request originates.
- *Service Provider* (SP): A mobile node containing a service available to other peer nodes and containing an Autonomic Node Manager (A-NM), defined in the previous Chapter.
- *Broker*: A mobile node managing the discovery and composition of a composition request and containing a Autonomic Composition Manager, defined in the previous Chapter.
- *Atomic Task*: An application level generated task, independent from any other task, requiring one or more atomic services. In the remainder of the thesis, the use of the word "task," shall imply an atomic task.

- *Atomic Service*: A service residing on a single SP, which can consist of further components if and only if they too reside on the same single SP. In the remainder of the thesis, the use of the word "service," shall imply an atomic service.
- *Task Composition Specification* (TCspec): A specification of service and service property requirements for a particular task, and information regarding the required execution flow of the desired composed service.
- *Service Composition Candidate* (SCC): A set of atomic services, which when composed satisfy the requirements of a particular TCspec.

A user's task (e.g. video conference call, music/video playback, Internet browsing) is represented as an TCspec. Explicitly included in a task's description are the services needed to meet the task, as well as optional constraints arising from supplementary capability requirements (e.g. video resolution). Moreover, additional constraints such as the specification of servicing location can be included. Service constraints exist as a means of satisfying both task requirements and user preferences.

5.4.1 QoS Function

In defining the QoS function, we consider important requirements of the virtual device concept, which include being aware of the user's continuously changing environment, instantly recognizing the expectations of the user, and instantiating a service with respect to the user's given circumstance. We form a QoS function in the context of virtual device composition, and define it as the degree of match between the requirements of a user's task (including service constraints satisfying both task requirements and user preferences) and the qualities and capabilities of service composition, taking into account the current state and availability of each service taking part in the composition. We formally represent this function as $\mathrm{QoS}_{\mathrm{vd}}$, where

$$QoS_{vd} = Max(QoS_{scc}), \tag{1}$$

$$QoS_{scc} = A_w \cdot \sum_i W_{ti,u} \cdot Sim_{ti}(S_{i,scc}, S_{i,TCspec}), \tag{2}$$

$$A_w = Min(A_{w,s}), \text{ and} \tag{3}$$

$$A_{w,s} = \det avail(ST_s, P_s, Q_s, ...). \tag{4}$$

Sim is a similarity function used to identify similarities in terms of qualities and capabilities between TCspecs and SCCs. More exactly, Sim_{Ti} is a specific similarity function for a particular service or service property S_i of type T_i. The output of the similarity function is a normalized value between [0,1], taking into consideration both positive and negative variations from an exact match, where the closer the value is to 1, the higher the degree of match. The summation index *i* runs through all matching services and properties of the TCspec and SCC in question. Applied to the similarity function in each iteration of the summation is a user-specific or default weight $W_{ti,u}$, representing the amount of weight given to a particular service or service property type. The whole of the summation is multiplied by a weight A_w representing the service availability of the potential service composition. A_w is obtained by computing the service availability $A_{w,s}$ for each service involved in the potential composition, and finding the minimum (i.e. Equation (3)). The computation of $A_{w,s}$ is based on various factors including the current state of the service ST_s, the usage policy of the service P_s, queue information Q_s regarding other users waiting for the service, and other information relevant to service availability (i.e. Equation (4)).

We are aware that the defining of similarity functions for each service and service property types is an exhaustive approach, such that potentially hundreds of types may exist. Moreover, in terms of real life applications, determining values for weights and the selection of the most appropriate similarity functions is a complex issue. However, we consider these issues out of the scope of this thesis, whereby we agree with [119] that solutions to such issues may be found through the investigation of machine learning techniques.

The remainder of this Section provides the details of how the above defined QoS function is applied to the various phases of the distributed service composition protocol.

5.4.2 Service Advertisement

Service advertisement is an ongoing process of the protocol throughout the four phases of service composition. In this process, SPs, using an advertisement message, periodically disseminate a list of their services to all nodes in radio range. Upon

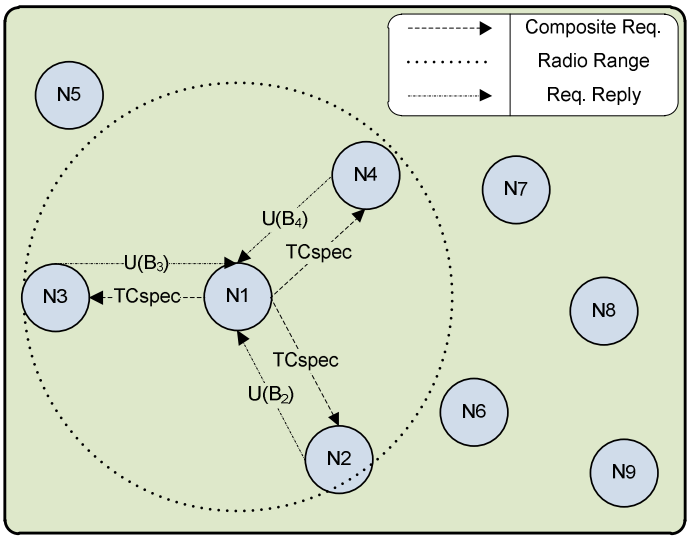


Fig. 5.1. Broker arbitration initiated by node N1 with a hop limit of 1.

receiving an advertisement message, a node stores information regarding the advertised service(s) in its service cache. We modify the advertisement message to include the service availability weights $A_{w,s}$, and respective service and service property values S_i for each service type T_i advertised in the message. This allows every node receiving the message to not only know the description of the advertised service, but in addition, become aware of the advertised service's current availability and specific T_i and S_i values for similarity function computation. To ensure cache freshness and validity of cached $A_{w,s}$, T_i, and S_i values, we attach to this information a short-spanning lifetime timer, which is carried into a node's cache. When the lifetime of a particular advertised service is up, the entry is deleted from a node's cache. In this manner, nodes only carry the most up-to-date information in their cache.

5.4.3 Broker Arbitration Phase

As the first phase of service composition, when a task becomes activated on a user's device, the respective node takes on the role of RS and constructs an TCspec specifying the services, qualities, and capabilities required by the task. Included in the TCspec are the service and service property weights $W_{ti,u}$ and respective S_i values for every service or service property type T_i involved in the activated task. In addition, the TCspec

contains execution flow information of the desired composed service. The RS embeds the TCspec into a composite request message, which it broadcasts to all nodes in its vicinity, as seen in Figure 5.1. As per the original protocol, when a node receives a composite request, it proceeds to compute its potential value, which it replies back to the RS. We modify the potential value function to include QoS_{vd} as a parameter. In doing so, we denote $U(B_i)$ as the utility value of each potential broker for a composite request as:

$$U(B_i) = S_S \cdot S_c(B_i) + W_L \cdot L(B_i) + W_J \cdot J(B_i) + W_{QoS} \cdot QoS_{vd}(B_i) \quad (5)$$

Where $S_c(B_i)$ is the number of service advertisements in B_i's cache, $L(B_i)$ is the battery life of B_i, $J(B_i)$ is the number of composite requests currently being processed by B_i, and $QoS_{vd}(B_i)$ is the result of the QoS_{vd} function computed by B_i. Applied to each parameter is a weight identifying the amount of influence given to a particular parameter during the computation of $U(B_i)$.

Each potential broker possesses sufficient information obtained through the modified service advertisement process to compute a QoS_{vd} value directly from its service cache. This value represents the quality of the virtual device it may potentially compose. From its cache, a potential broker is able to directly form multiple SCCs. Every SCC is compared against the TCspec supplied by the RS via Equation (2). Using Equation (1) the QoS value of the potential broker's top ranked SCC (i.e. QoS_{vd}) is utilized within the computation of $U(B_i)$. The elected broker is the node, which maximizes $U(B_i)$. In this manner, not only will the elected broker have sufficient resources to perform composition, but, in addition, will be located in a high service density area best suited to form a virtual device with the highest QoS currently available in the user's vicinity.

5.4.4 Service Discovery Phase

As the third phase of composition, the elected broker first computes a fresh cached-based QoS_{vd} value to account for the effect of any mobility or topology changes since broker arbitration. Although some changes may have occurred in the network, we do not anticipate significant differences altering the results of broker arbitration; however, we do consider the possibility. To counter this effect, we reasonably assume that the results of broker arbitration accurately represent the task-specific QoS currently available in the

user's vicinity, and introduce a dynamic threshold *Td*, which, after every broker arbitration is set to the winning broker's QoS_{vd} value. Following broker arbitration, the winning broker compares its fresh QoS_{vd} to *Td*, and if fresh $QoS_{vd} \geq Td$, we allow the broker to attempt to compose a service directly from its cache information without further discovery effort. Should fresh $QoS_{vd} < Td$, the broker proceeds to discover additional services using GSD, such that the broker's cache no longer represents the QoS currently available in the user's vicinity. However, the nodes potential as a broker remains valid such that it is likely that the desired services, although now out of advertisement range, are still relatively close. We modify the GSD protocol slightly to have every SP receiving a discovery request message, matching its provided service, include as part of the reply the respective service availability weights $A_{w,s}$, and service and service property values S_i for each service type T_i it provides.

5.4.5 Service Integration and Execution Phases

In the final phases of composition, the broker integrates the discovered services forming the top ranked SCC as per equations (1-4). As a result of the application of the QoS function and modifications to the preceding phases of the protocol, the final composed service represents the best possible virtual device, satisfying the user task requirements, currently available within the vicinity. We do not modify any aspect of the execution phase, whereby execution proceeds as specified in the original distributed protocol.

5.5 Simulation Results and Analysis

This section evaluates the performance of our distributed protocol for virtual device composition (DVDC) for different composite lengths in varying service densities, mobility, and topologies. We compare our results to those of the original protocol (DSC), in an effort to identify the gain in quality of service and the respective costs in terms of composition time and number of messages needed.

Duration	100 consecutive requests
Space(x,y)	190×190 m
Transmission range	30 m
No. of nodes	64
Advertisement interval	5 sec
Advertisement lifetime	7 sec
Control hop count	1
Mobility	Random-way-point model with a minimum speed of 1m/s, a maximum speed of 3m/s, and a 5s stoppage time
Initial topology	Grid topology with nodes equally spaced out in (x,y)
Composite lengths	3, 5, and 7
Service density	20 to 100%

Table 5.1. Simulation parameter summary.

5.5.1 Simulation Parameters and Metrics

We build a MANET and implement the distributed service discovery protocol, including GSD, and our extension using J-SIM [120], a well known simulation environment. We utilize an area of 190x190m containing 64 nodes with a transmission range of 30m. We set mobility to follow a random-way-point model with a minimum speed of 1m/s, a maximum speed of 3m/s, and a 5s stoppage time. All SPs follow a 5s advertisement interval, where all advertised services carry a lifetime of 7s. All broadcasts used by the broker arbitration phase and GSD follow a strict control hop count of 1. Our experimentations reflect a mobile device entering the simulation environment and making 100 consecutive requests. The experiment is repeated for composite lengths 3, 5, and 7, and service densities 20-100%. We define service density as the percentage of nodes containing one of the atomic services required in the composition. Identified in each experiment is a QoS value representing the degree of match between the QoS_{vd} value of the resulting composed service and the QoS_{vd} value of the best possible service composition currently available in the user's vicinity. Moreover, identified in each experiment is the amount of time and number of messages consumed.

5.5.2 QoS Analysis

As previously mentioned, the QoS metric used represents the degree of match between the QoS_{vd} value of the resulting composed service and the QoS_{vd} value of the best possible service composition currently available in the user's vicinity. Figure 5.2 shows the effect of service density on QoS for various composite lengths. Results indicate that DVDC provides an average increase in QoS of 49%, 43%, and 39% when composing 3, 5, and 7 services, respectively, with QoS staying above 90% regardless of service density. As service density increases, DVDC performs better, due to the increase of service knowledge within the network. On the contrary, for the same reason, the original protocol performs worse, such that no mechanism exists to identify a worse service from a better service, leading to a performance that follows the average QoS of the network, which drops as service density rises.

5.5.3 Cost Analysis

Figures 5.3 and 5.4 depict the cost of DVDC with regards to messages consumed and composition time, respectively. As expected, the number of messages used by DVDC is higher than that of the original protocol by an average of 29, 34, and 31 messages when composing 3, 5, and 7 services, respectively. Moreover, an expected average time increase of 3.28, 2.80, and 3.83 seconds, respectively, is also present. These costs are affected by mobility, and arise from the additional work required to discover the best service among a collection of services of the same type; functionality not present in the original protocol. Obviously, the higher the service density, the higher the amount of work that needs to be done. However, the amount of time used remains relatively stable, due to the use of predetermined timeout values for election and discovery. Affecting time and number of messages is changing network topology leading to sparse network regions and even the isolation of the requesting source. Such scenarios lead to failing and repetitive broker elections. Furthermore, Figure 5.4 shows that DVDC leads to a lesser time cost in lower service densities for higher composite lengths. This can be attributed to the fact that DVDC considers the results of broker arbitration as accurately

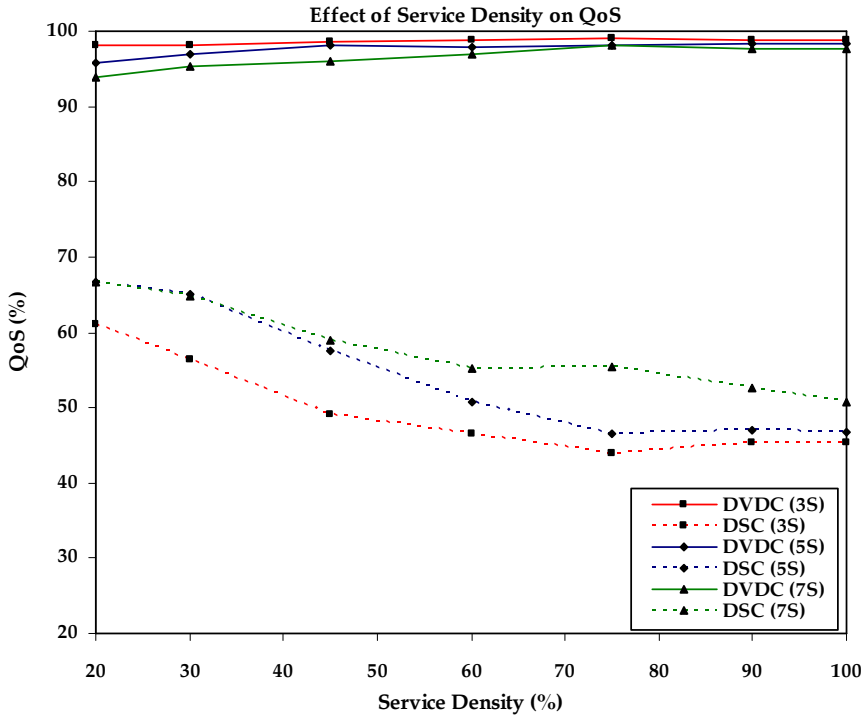


Fig. 5.2. QoS with respect to service density for composition lengths 3, 5, 7.

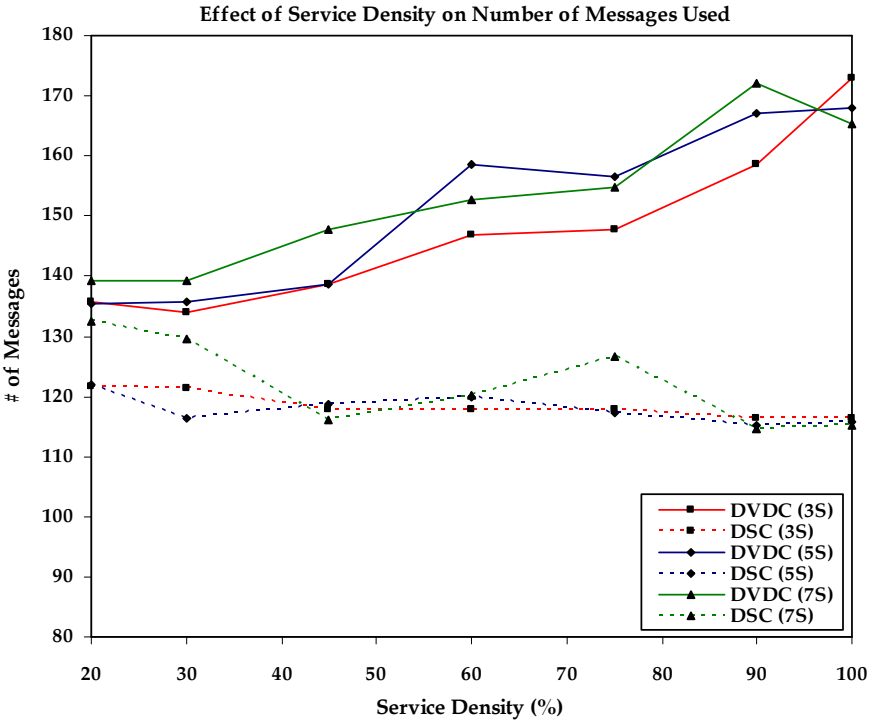


Fig. 5.3. Number of messages consumed with respect to service density for composition lengths 3, 5, 7.

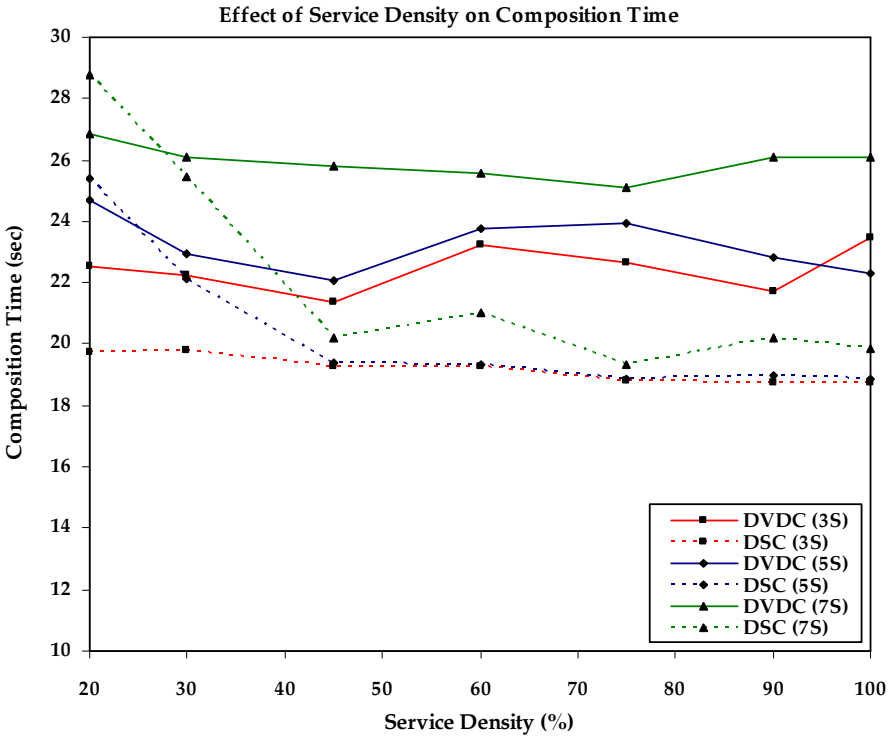


Fig. 5.4. Composition time with respect to service density for composition lengths 3, 5, 7.

representing the amount of QoS currently available in the user's vicinity. Therefore, when broker arbitration indicates a particular required service as unavailable, no further attempt is made to locate it, whereas, the original protocol is unable to make this assumption, and does utilize time to make further attempts in the discovery phase.

It should be noted, that the significant cost of periodic broadcasting, required by both DVDC and DSC were omitted during analysis to allow for a more accurate comparison, hence the low message cost numbers. Experiments in the following section do not omit this cost.

5.5.4 Effects of Scaling and Mobility

In investigating the effect of scaling on the cost of the two protocols, we hold the service density at 50% and the composition length at 3 services. Mobility continues to follow a random-way-point model with a minimum speed of 1m/s, a maximum speed of

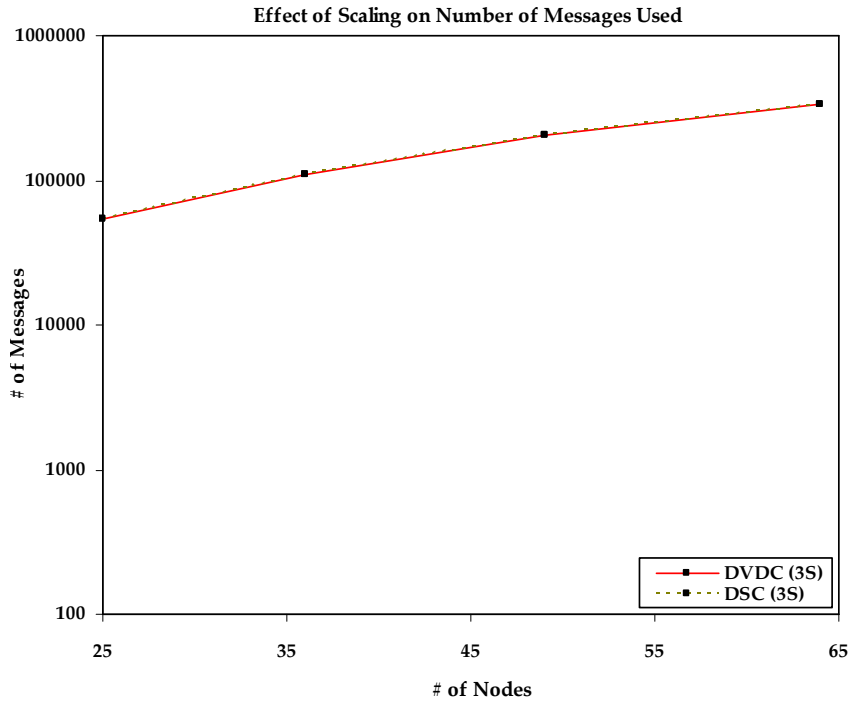


Fig. 5.5. Number of messages consumed with respect to scaling number of nodes and dimension.

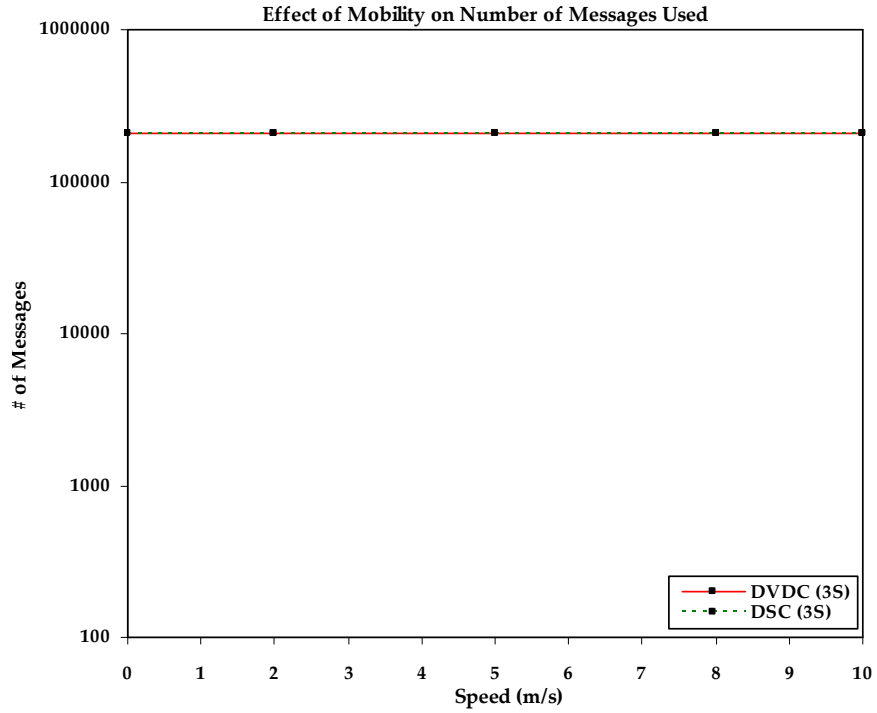


Fig. 5.6. Number of messages consumed with respect to mobility speed.

3m/s, and a 5s stoppage time. All broadcasts remain at a strict bounded hop count of 1 and service advertising intervals are set to 5s. However, with a node's transmission range holding at 30m, we gradually increased the dimension area *D* of the experiment as well as the number of nodes *N* involved, such that $D = 30 + 6 \times (\sqrt{N} - 5)$.

Results show that both protocols suffer from scalability issues. The performance in terms of QoS and composition time of both DVDC (99% and 22s) and DSC (50% and 19s) remain unchanged due to the continuous broadcasting and heavy discovery mechanism in the protocols. However, the cost in messages places a burden on the network that is far too heavy to be practical. Figure 5.5 (curves overlap) shows the number of messages required, including those used in service advertisement. The benefit of either protocol is now overshadowed by the sheer number of messages in the advertisement process. The messages required in the arbitration and service discovery are no longer significant. This cost grows as the dimension and number of nodes increases.

To examine the effects of mobility, we hold the service density at 50%, the composition length at 3 services, the number of nodes at 49, the dimension at 42x42m, and the transmission range at 30m. Hop count remain at 1 and service advertising at 5s. Mobility continues to follow a random-way-point model, but now gradually increases, with no stoppage time, from 0m/s to 10m/s.

Results show the broadcast-based discovery mechanism is resilient to mobility; for both DVDC (99% and 22s) and DSC (50% and 19s) QoS and composition do not change. However, the cost far outweighs the benefit; once again the number of messages (Figure 5.6, curves overlap) required places a far too heavy burden on the network to be practical.

This leads to the conclusion that further work should be conduced to minimize the use of broadcast-based service advertisements, as this causes congestion and the unnecessary depletion of node resources such as battery and processing power.

5.6 Summary

In this chapter, we have presented an extension to a prominent broker-based service composition protocol for MANETs, allowing for a service composition that forms the best possible virtual device at the time of need. In doing so, we embed and apply in a distributed manner a QoS function, which identifies the variation of similarity between two services of the same type, the current availability of each service, and how important such variation and availability is to the user. Altered, were the various phases of composition with an emphasis on electing a broker that not only has sufficient resources to perform composition, but is located in a high service density area best suited to form a virtual device with the highest QoS currently available in the user's vicinity. Moreover, the protocol remains decentralized, load-balancing, and fault tolerant. We have compared the performance of the extended protocol against that of the original. Simulation results show an 81% average increase in QoS at the cost of a 26% average increase in messages and 17% average increase in composition time.

Although results show a high increase in QoS at a low cost in terms of increased amounts of messages and composition time, our extension relies on a discovery mechanism largely based on constant periodic service advertising by controlled broadcast, whether composition is currently desired or not. The effect of which, is the unnecessary depletion of node resources (i.e. battery and processing power). In the following Chapter we examine the possibility of performing virtual device composition in MANETs without the use of broadcast-based service advertisements.

Chapter 6

A Distributed Constraint Satisfaction Problem for Virtual Device Composition

In the previous chapter, we provided a distributed service composition protocol tailored for virtual device composition. However, similar to current MANET-based composition schemes, the protocol makes use of service discovery mechanisms dependent on periodic service advertising by controlled broadcast, resulting in the unnecessary depletion of node resources (i.e. battery and processing power). Moreover, the scheme does not allow for the protection of private or security sensitive information in the process of composition. In this Chapter, we present a distributed constraint satisfaction problem for virtual device composition in MANETs addressing these issues, and through simulation show its high efficiency and QoS-awareness.

6.1 Introduction

As mentioned in the previous chapter, an open problem in achieving virtual devices is how to allow for the automatic composition of devices, and the services they provide, to create dynamic service configurations with limited human intervention, where the aim is

to automate the discovery process and enable devices to determine what services are required. The hope of finding more reactive and decentralized solutions providing support for mobile ad hoc network environments (MANETs) has lead to the emergence of research work specifically targeting service composition in MANETs (e.g. [65], [112], [118]). However, such works rely on discovery mechanisms largely based on constant periodic service advertising by controlled broadcast, whether composition is currently desired or not. The effect of which, is the unnecessary depletion of node resources (i.e. battery and processing power). Furthermore, such works require that all information necessary in deciding composition be divulged, a less than favorable act, if this information is sensitive with regards to privacy and security.

In the previous Chapter, we presented an extension to D. Chakraborty *et al.*'s dynamic broker-based distributed service composition protocol [65] for MANETs. Although results showed a high increase in quality of service at a low cost in terms of increased amounts of messages and composition time, our extension still relies on continuous service advertising by controlled broadcast and does not address the issue of divulging secure/private information. In this Chapter we examine the possibility of performing virtual device composition without the use of broadcast-based service advertisements, by modeling and solving the virtual device problem as a *distributed constraint satisfaction problem (distCSP)* [116]. A CSP is defined by the triplet *(X, D, C)*, where $X = \{X_1,..., X_n\}$ is a set of *n* variables, $D = \{D_1,..., D_n\}$ is a set of *n* domains of values, such that D_i is the domain of the values of the variable X_i, and $C = \{C_1,..., C_j\}$ is a set of *j* constraints of the problem. A CSP is solved by assigning values to variables whereby all constraints are satisfied. A distributed CSP is defined as a CSP where variables and constraints are distributed among automated agents, such that solving a CSP implies achieving coherence or consistency among agents.

Another benefit of approaching the problem as a distCSP, is that unlike traditional service composition schemes, *distCPSs can be solved without the need for agents to directly divulge neither complete nor precise information about their domain and constraints*. Instead, information is passed in a highly summarized form. This is relevant to virtual device composition for privacy and security reasons, such that there may be

information pertinent to composition that a node may not wish to divulge (e.g. client billing model).

The resulting contributions of this Chapter are a distCSP model for task-based, quality-of-service aware, virtual device composition in MANETs, and a virtual device distCSP protocol utilizing an asynchronous backtracking-based algorithm for solving the respective distCSP. Through simulation and comparison we show the effectiveness of our method.

The rest of this Chapter is organized as follows. Sections 6.2 and 6.3 present the problem formulation and model. Section 6.4 describes our protocol and asynchronous backtracking-based algorithm. Section 6.5 presents our simulation results, and Section 6.6 gives a summary.

6.2 Problem Formulation

Once again, we consider an environment composed of mobile nodes, connected together using ad hoc network protocols, representing various devices providing single or multiple services, able to be invoked by peer nodes. Device composition involves the composition of services offered by devices in a user's vicinity and provides a virtual device to the user, satisfying an atomic task. We define an *atomic task* as an application level generated task, independent from any other task, requiring one or more atomic services. We define an *atomic service* as a service residing on a single node, which can consist of further components if and only if they too reside on the same single node. Generated from a task is a *task-based composition specification (TCspec)*, which is a specification of service and service property requirements, represented as policies. Policies are divided into three categories: task requirement, service constraint, and user preference. *Task requirements* represent the minimum services required for task satisfaction. *Service constraints* represent the necessary constraints on the services to allow the composed service to function as the application intended. *User preferences* are service constraints controlled by the user, which are present solely to shape the user's desired quality of Service (QoS). We refer to QoS, such that at any instant, numerous potential compositions exist in the environment satisfying a task.

Separating one composition from another is the QoS associated with the particular composition.

Our objective is to maximize the QoS generated by a composed virtual device, by efficiently examining possible composition configurations in a user's vicinity and identifying the configuration that satisfies all service requirements, adheres to all service constraints, and best caterers to the users' identified preferences. Complicating this process is the fact that nodes are independent and may be operating under different sets of rules and guidelines used in making composition decisions. We refer to such rules and guidelines as private policy, where *private policy* is information used by a node, in addition to *public policy* (i.e. policies contained in a TCspec), to decide how to react towards satisfying tasks. Private policy may refer to security rules, business models, priority guidelines, and etc. Information regarding private policy is confidential and should not be shared with other nodes. This makes centralized solutions or methods involving high amounts of information sharing unsuitable for solving this problem. An acceptable solution is a distributed one, utilizing minimal information sharing and providing efficient negotiation techniques.

6.3 Problem Modeling

In satisfying the objective outlined in the previous Section, we model virtual device composition in the form of a distributed constraint satisfaction problem *VD-DistCSP* $= (X, D, C)$ where:

- $X = \{X_i\}(i = 1,...,n; X_i \subseteq D)$. X_i is a variable corresponding to the set of services being provided by node *i* to satisfy a particular task.
- $D = \{D_i\}(i = \{1,...,n\})$. *D* is a set of *n* domains of values, such that D_i is the service option domain of X_i.
- *C* is the set of constraints of the problem. We can formulate them as follows:

$$\forall_{i,j} \quad X_i \cap X_j = \{\ \}\ \ i, j \in \{1,...,n\}, i \neq j \tag{1}$$

$$\forall_i \quad \bigcup X_i = \{S\} \tag{2}$$

S is a set of services corresponding to all *k* required services d_j, $j = \{1,...,k\}$, where each service corresponds to a particular task requirement policy of a TCspec. Constraint (1) ensures that at most one node provides any particular service. Constraint (2) guarantees that all required services have been assigned. Furthermore, the provision of any service by a node is additionally constrained by that node's distinct private policy. Moreover, service constraints and user preferences are not used as direct constraints on the problem, but instead are used to identify QoS.

Similarly to D. Chakraborty *et al*.'s solution, we utilize a broker, whereby in electing a broker we denote a utility function $U(B_i)$ as the utility value of each potential broker as:

$$U(B_i) = W_L \cdot L(B_i) + W_{CR} \cdot CR(B_i) + W_P \cdot P(B_i) \quad (3)$$

Where $L(B_i)$ is the battery life of B_i, $CR(B_i)$ is the number of composite requests currently being processed by B_i, and $P(B_i)$ is the processing power of B_i. Applied to each parameter is a weight identifying the amount of influence given to a particular parameter during the computation of $U(B_i)$.

In identifying the QoS of a particular composition we apply the QoS_{vd} model defined in Section 5.4.1, which the provide bellow for reference:

$$QoS_{vd} = Max(QoS_{scc}), \quad (4)$$

$$QoS_{css} = A_w \cdot \sum_j W_{t,u} \cdot Sim_t(S_{j,scc}, S_{j,TCspec}), \quad (5)$$

$$A_w = Min(A_{w,s}), \text{ and} \quad (6)$$

$$A_{w,s} = \det_{avail}(ST_s, P_s, Q_s, ...). \quad (7)$$

Having identified our constraint satisfaction problem, we apply an algorithm based on the asynchronous backtracking algorithm [116] to solve the virtual device composition problem in MANETs. We call our method the *virtual device distributed constraint satisfaction problem* (VD-DistCSP) protocol and our respective algorithm used in the protocol the *virtual device backtracking* (VD-Btracking) algorithm.

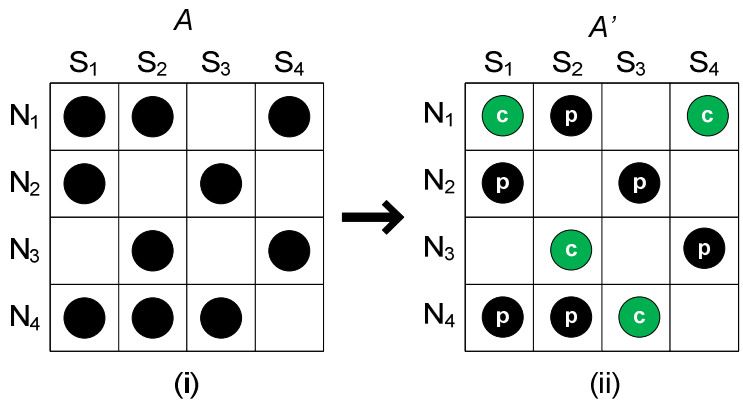


Fig. 6.1. A transformation from Matrix A to A', visually abstracting the virtual device problem. (c = committed, p =potential).

6.4 VD-DistCSP Protocol

The virtual device distributed constraint satisfaction problem (VD-DistCSP) protocol allows automated agents (i.e. A-CM and A-NMs), each representing the interests of a particular node, to act concurrently and asynchronously without any global control, in deciding how to form the best possible virtual device at the time of need, with regards to the user's task-based desires. The protocol does not require periodic service advertising and provides a general solution for incorporating private policy.

The virtual device composition problem, from a distCSP perspective can be represented as an $m \times n$ Matrix A (Figure 6.1.i), comprised of m nodes and n services, such that:

- A black circle in $A_{x,y}$ represents a service $d = S_y$, such that N_x provides service d.

The solution to the virtual device composition problem, requires the application of the constraint set C defined in Section 3, whereby a Matrix $A'_{x,y}$ (Figure 6.1.ii) is formed such that:

- A black circle in $A_{x,y}$ containing the letter p represents a *potential* service $d = S_y$ in a node N_x's X_x variable, such that N_x has the potential of contributing service d in the virtual device composition.

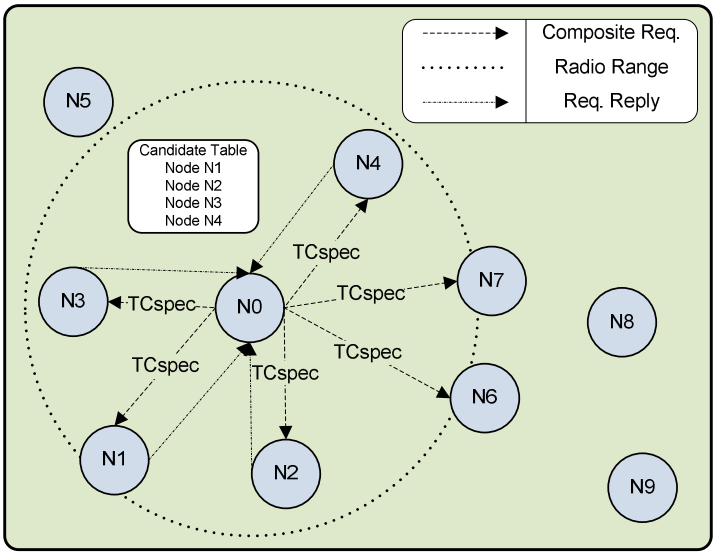


Fig. 6.2. A sample candidate formation scenario.

- A black circle in $A_{x,y}$ containing the letter *c* represents a *commited* service $d = S_y$ in a node N_x's X_x variable, such that N_x has committed to contributing service *d* in the virtual device composition.
- Each column *n* must contain *exactly one* black circle containing the letter *c*.

The composition of node/service pairs corresponding to the black circle values containing the letter *c* form a composition satisfying the constraint set *C*.

The VD-DistCSP protocol establishes a Matrix *A* and provides a concurrent and asynchronous negotiation mechanism to transform Matrix A into Matrix A'. The following presents the various steps of the VD-DistCSP protocol.

6.4.1 Candidate Formation

Candidate formation is the first step of the protocol and allows for the creation of Matrix *A*. When a task becomes activated on a user's device, the respective node sends out a *virtual device request (VDrequest)* message into the user's immediate vicinity, using a bounded broadcast (Figure 6.2). A VDrequest contains the task's TCspec, which is used by a receiving node to decide whether or not it can contribute a service towards achieving task requirements. A node able to contribute to a task, sends a *virtual device reply (VDreply)* message. The VDrequest initiator utilizes these replies to compute a

candidate table containing the addresses and unique identifiers of all responding nodes. The nodes in the candidate table are organized in *descending priority order* based on their unique identifiers. Once the table is formed, the VDrequest initiator distributes the candidate table to all respective candidates.

Figure 6.2 shows an example of candidate formation. A task has become activated at Node N_0, at which time it broadcasted, with bounded value of one hop, a VDrequest containing the respective TCspec. Nodes N_1, N_2, N_3, and N_4 determine that they are candidates, such that they provide one or more services listed in the TCspec, and reply with a VDreply message. Using the VDreply messages, N_0 forms a prioritized candidate table and distributes the candidate table to all respective candidates.

6.4.2 QoS Ranking

A node receiving a candidate table implies its participation in the VD-Btracking protocol. For each service d_j that such a node N_i has agreed to potentially contribute, a *QoS ranking* is computed using the principles of Equation (5) as $Rank_{qos} = A_{w,s} \cdot W_{t,u} \cdot Sim_t(d_j, d_{j,TCspec})$, where a similarity function is computed between d_j and the equivalent service requirements specified in TCspec. Applied to the similarity function are the availability and user-specific/default weights $A_{w,s}$ and $W_{ti,u}$, respectively, of the service d_j.

For each service a node has agreed to potentially contribute, three pieces of information are kept in a tertiary: service type, QoS ranking, and commitment status. The *commitment status* takes the value *potential* (*p*) or *committed* (*c*) and represents whether the service has been committed to composition or not, which reflects *p* and *c* values shown in Figure 6.1.ii. For each of a node N_i's such services a tertiary is stored in N_i's X_i, such that $X_i = \{(d_j, Rank_{qos}, ComStatus_j)\}(j = \{1,...,k\})$.

Figure 6.4.a represents a global view of the Matrix *A* formed following the candidate formation in the previous section. Included are the respective QoS rankings. Looking at node N_1, its $X_1 = \{(S_1,100,c),(S_2,95,c),(S_4,100,c)\}$.

```
when received (ok?, [N_i,X_i]) do – (i)
  add[N_i,X_i] to agent_view;
  check_agent_view;
end do;

when received (nogood, N_i, nogood) do – (ii)
  add nogood to nogood_list;
  check_agent_view;
end do;

procedure check_agent_view – (iii)
  when agent_view and X_self are not consistent do
    if no value in D_self is consistent with agent_view then
    backtrack;
    else
      select d∈D_self where agent_view and d are
      consistent;
      X_self ← d;
      send (ok?,(N_self, X_self) to candidates;
    end if;
  end do;

procedure backtrack – (iv)
  nogoods ← {V|V = inconsistent subset of agent_view};
  when an empty set is an element of nogoods do
    broadcast to all agents that there is no solution,
    terminate this algorithm;
  end do;
  for each V ∈ nogoods do
     send (nogood, N_self, V) to N_j∈V
  end do;
```

Fig. 6.3. VD-Btracking algorithm pseudocode.

6.4.3 VD-Btracking Algorithm

Now that Matrix *A* is complete, we apply the virtual device backtracking (VD-Btracking) algorithm, which is based on the traditional asynchronous bactracking algorithm for solving distCSPs [116]. The execution of the VD-Btracking algorithm (pseudocode provided in Figure 6.3) is asynchronous between candidate nodes and begins with all candidates committing to all services in their respective X_i's and exchanging these values using *ok?* messages.

Ok? messages are used to spread knowledge of value assignments to candidate nodes, which candidate nodes use to form their *agent view*. An agent view reflects a node's current view of the *partial solution*. A partial solution is a subset of the *final solution*,

which is expanded by adding committed variable tertiaries one by one, until a complete and final solution is reached. Different nodes may have different agent views at any given time, although as the algorithm progresses, all nodes work towards a single common agent view (i.e. final solution).

Following the receipt of an *ok?* message (Figure 6.3.i), a node N_i compares its X_i with its agent view (Figure 6.3.iii), looking for any inconsistencies with regards to *higher priority nodes*. Inconsistencies occur when N_i's X_i value violates the set of constraints C of the problem. If inconsistencies are present, N_i attempts to alter its X_i to resolve the inconsistencies. If N_i is able to achieve consistency, it distributes its new X_i in an *ok?* message to all candidates. If no such X_i exists, then it is necessary to *backtrack* (Figure 6.3.iv). Backtracking involves the sending of one or more *nogood* messages to *higher priority nodes* causing the inconsistency. A *nogood* message specifies to the receiver exactly what tertiary value in X_i to consider as *nogood*. Every candidate keeps a *nogood list*, which adds to its constraint set C. Upon receiving a *nogood* message, the recipient attempts to adapt its respective X_i value, enabling the sender to re-enter a consistent state (Figure 6.3.ii). If the *nogood* recipient is unable to resolve the inconsistency, then it too triggers a backtrack. In this way, *nogood* messages flow up the candidate table until they are either resolved, or a no solution decision is reached. A no solution occurs when the highest priority node cannot resolve a *nogood*, such that it has exhausted all of its variable options. This leads to the termination of the algorithm by failure as a result of the problem being over constrained.

A solution presents itself when all nodes share a common agent view and are in a consistent state. Nodes continue to periodically send *ok?* messages until either a no solution or final solution is reached.

6.4.4 Example Execution

In clarifying the algorithm we provide a sample execution continuing our current example. It should be noted that the trace of the algorithm's execution can vary according to timing/delay of messages, and that this example shows one possible trace of execution.

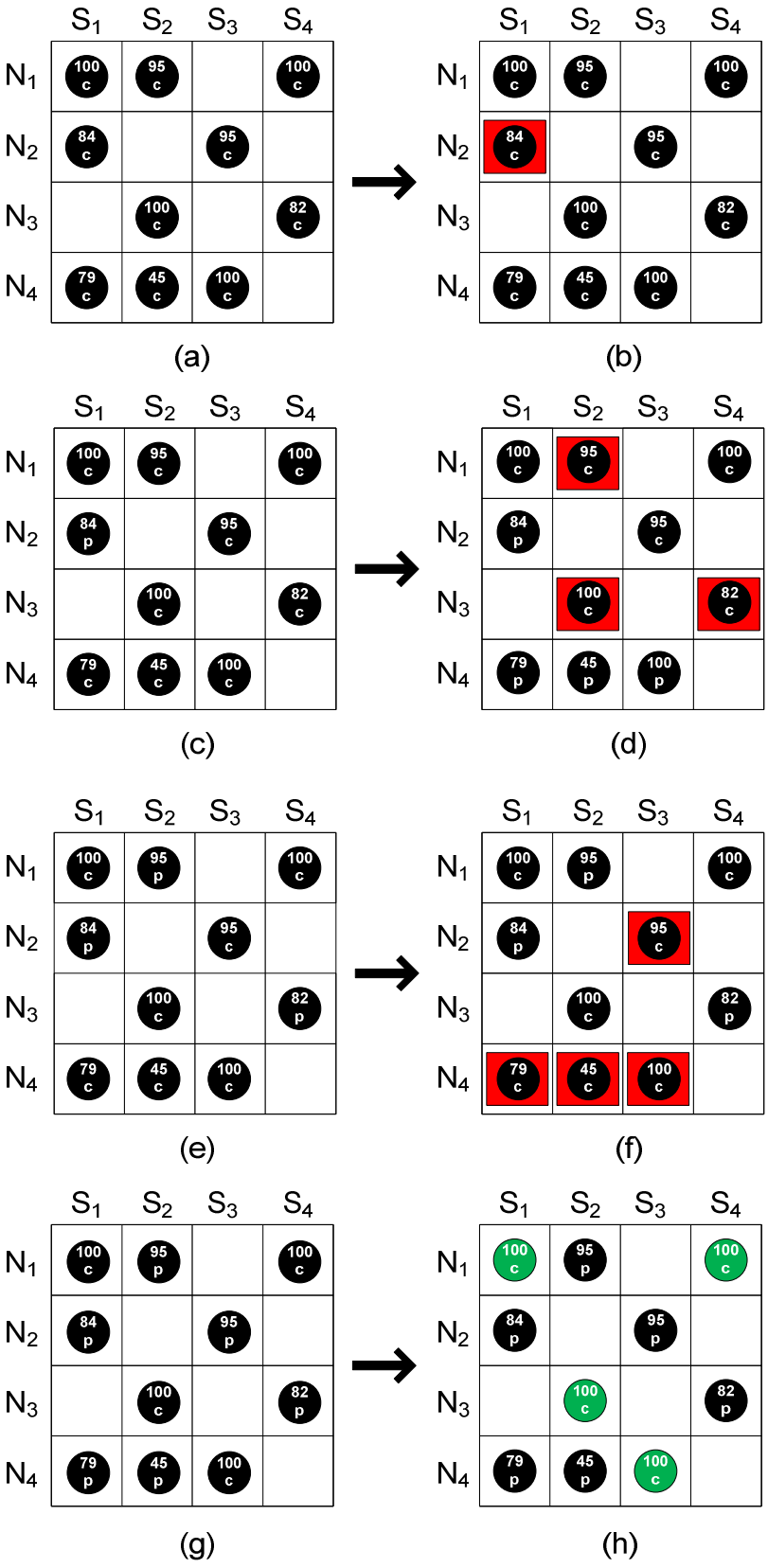


Fig. 6.4. Example execution of the VD-Btracking algorithm.

The initial values are shown in Figure 6.4.a, where priority has been determined by numerical identifiers, QoS rankings have been computed, all candidates have committed to their services, and these values have been exchanged using *ok?* messages. Each candidate now attempts to reach consistency.

N_1, being the highest priority node, is currently in a consistent state; however, N_2 is not. N_2 observes N_1 as having committed to service S_1 (Figure 6.4.b). In order to enter a consistent state N_2 must un-commit to S_1, such that it does not provide a higher QoS ranking. N_2 keeps its commitment to S_3, whereby N_1 does not provide service S_3 (Figure 6.4.c).

N_3 is also in an inconsistent state (Figure 6.4.d). N_3 is able to rectify its S_4 value by un-committing; however, since its QoS ranking for S_2 is higher than that of N_1, it is unable to un-commit to S_2. Therefore, N_3 sends a *nogood* message to N_1. N_1 receives the *nogood* message, and un-commits to S_2, allowing N_3 to enter a consistent state (Figure 6.4.e).

N_4's values for S_1, S_2, and S_4 are inconsistent. N_4 un-commits to services S_1 and S_2, but is unable to un-commit to S_3 (Figure 6.4.f). It sends a *nogood* to N_2, which allows N_4 to reach consistency (Figure 6.4.g), and a solution is found (Figure 6.4.h).

6.4.5 Algorithm Soundness and Completeness

The soundness of the VD-Btraking is guaranteed such that agents can only reach a stable state if and only if all of their variable values satisfy all constraints. The algorithm is also complete, whereby the algorithm finds a solution if one exists and terminates if there is no solution, in finite time.

A no solution scenario occurs if the problem is over constrained. The VD-Btracking responds to over constrained problems with the eventual generation of a *nogood* depicting an empty set, such that any set of variable values leads to a constraint contradiction. This implies a failure, causing the termination of the algorithm. The only way that the termination of the VD-Btracking algorithm would not happen in finite time is if one or more nodes were to cycle through their possible variable values in an infinite loop. By induction we show that this cannot happen.

Assuming the base case where the node of highest priority is in an infinite loop, we consider the following. Since the node is of highest priority, it will not send *nogood* messages, but will only receive them. When the node receives a *nogood*, it will change its value such that it either receives a *nogood* in return or does not receive a *nogood* in return. Should the node receive a *nogood* for all of its potential values, an empty set *nogood* will be generated. Should the node not receive a *nogood* after changing its variable value, then no further variable changes will occur. Either way, an infinite loop cannot occur.

Now, we assume that nodes X_1 to X_{k-1} (k>2) are in a stable state, and that node X_k is in an infinite loop. Since X_1 to X_{k-1} are in a stable state and X_k only receives *nogood* messages from lower priority nodes, this implies that all *nogood* messages received should be resolved by a change in X_k's variable value. Due to the fact that X_k's potential variable values are finite, two situations can occur: either X_k changes it value and does not receive a *nogood* in return, or it exhausts its potential variables values and sends a *nogood* to $X_{1\ldots k-1}$. The first situation presents a contradiction such that X_k would not be in an infinite loop, and the second situation presents a contradiction such that $X_{1\ldots k-1}$ cannot receive a *nogood* if they are in a stable state. Therefore, X_k cannot be in an infinite loop and must terminate in finite time.

With regards to complexity, constraint satisfaction is generally considered NP-complete [116]. As a result, worst-case time complexity is exponential in the number of variables n, $X = \{Xi\}(i = 1,...,n; Xi \subseteq D)$.

6.5 Simulation Results and Analysis

This section evaluates the performance of our VD-DistCSP protocol for different composite lengths in varying service densities, mobility, and topologies. We compare our results to those of the distributed protocol for virtual device composition (DVDC) from the previous Chapter, as well as the original protocol (DSC) from the works of Chakraborty *et al*. [65], which DVDC extends, in an effort to identify the efficiency gain in terms of message usage and composition time.

6.5.1 Simulation Parameters and Metrics

We build a MANET and implement the VD-DistCSP protocol using J-SIM [120], a well known simulation environment. We utilize an area of 30x30m containing 25 nodes with a transmission range of 30m, in an attempt to simulate a home environment. We set mobility to follow a random-way-point model with a minimum speed of 1m/s, a maximum speed of 3m/s, and a 5s stoppage time. All broadcasts follow a strict bounded hop count of 1 and the DVDC and DSC service advertising intervals are set to 5s. Our simulations reflect a mobile device entering the simulation environment and making 100 requests over a 24 hour period. The simulation is repeated for composite lengths 3, 5, and 7, and service densities 20-100%. We define service density as the percentage of nodes containing one of the atomic services required in the composition. Identified in each simulation is the achieved QoS with respect to the best QoS currently available, and amount of time and number of messages consumed.

Duration	100 requests over 24 hour period
Space(x,y)	30×30 m
Transmission range	30 m
No. of nodes	25
Advertisement interval (DVDC and DSC)	5 sec
Advertisement lifetime (DVDC and DSC)	7 sec
Control hop count	1
Mobility	Random-way-point model with a minimum speed of 1m/s, a maximum speed of 3m/s, and a 5s stoppage time
Initial topology	Grid topology with nodes equally spaced out in (x,y)
Composite lengths	3, 5, and 7
Service density	20 to 100%

Table 6.1. Simulation parameter summary.

6.5.2 QoS Analysis

As previously mentioned, the metric used in measuring QoS represents, as a percentage, the degree of match between the composed QoS and the best QoS currently available in the network. From Figure 6.5, results indicate that VD-DistCSP and DVDC provide an average increase in QoS, from DSC, of 55%, 44%, and 35% when composing 3, 5, and 7 services, respectively. VD-DistCSP and DVDC are both able to provide QoS above 99% for all service densities in Figure 6.5. DSC does not fair well, mainly due to the fact that it contains no mechanism to identify a worst service from a better service. Therefore the performance of DSC, in terms of QoS, simply follows the average QoS of the networks, which drops as service density rises.

6.5.3 Cost Analysis

Averaged to per single composition, the number of messages used by VD-DistCSP (see Figure 6.6) is significantly lower than that of DVDC and DSC by an average of 62,750 messages (98%), such that the DVDC and DSC curves overlap in Figure 6.6. Furthermore, an average composition time decrease by VD-DistCSP of 9.68 seconds (64%) and 15.39 seconds (74%) is present when compared to DVDC and DSC, respectively (see Figure 6.7), such that DVDC holds at about 15 seconds for all service densities shown in Figure 6.7.

The considerable drop in messages arises from DVDC and DSC requiring nodes to advertise their services at periodic intervals to ensure up-to-date service knowledge in the network. Moreover, DVDC and DSC require the sequential execution of broker election, service discovery, and service aggregation phases, whereby VD-Btracking solves the composition in parallel as a single distCSP, taking less time. Furthermore, the initial spike of DSC for composition lengths 5 and 7 in Figure 6.7, can be attributed to the fact that DSC makes futile attempts to discover services that do not exist, whereby VD- DistCSP and DVDC, in finding the ‘best’ composition possible, are able to quickly identify a desired service as existing or not.

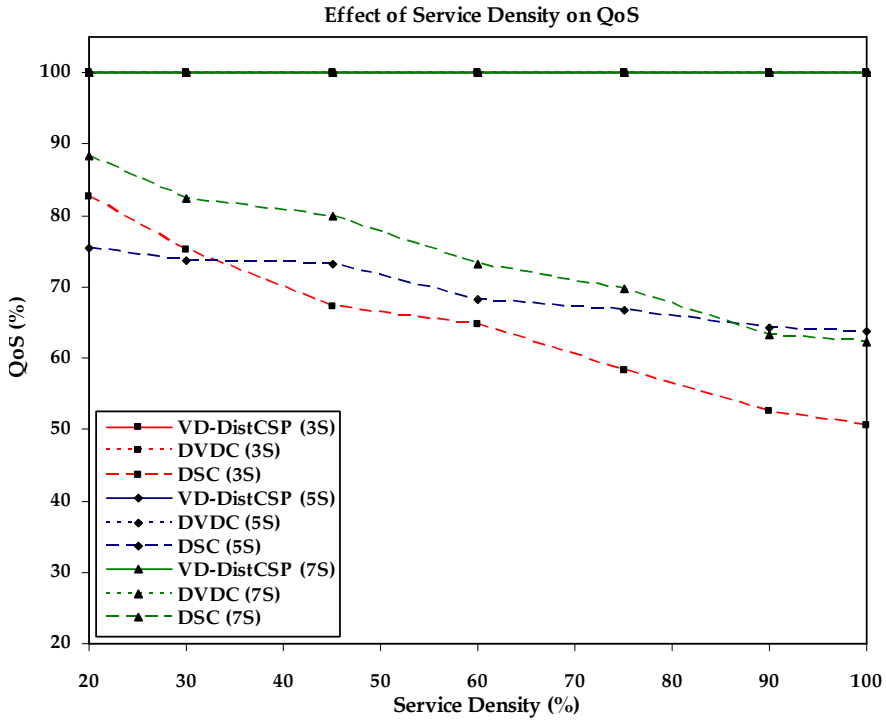


Fig. 6.5. QoS with respect to service density for composition lengths 3, 5, 7.

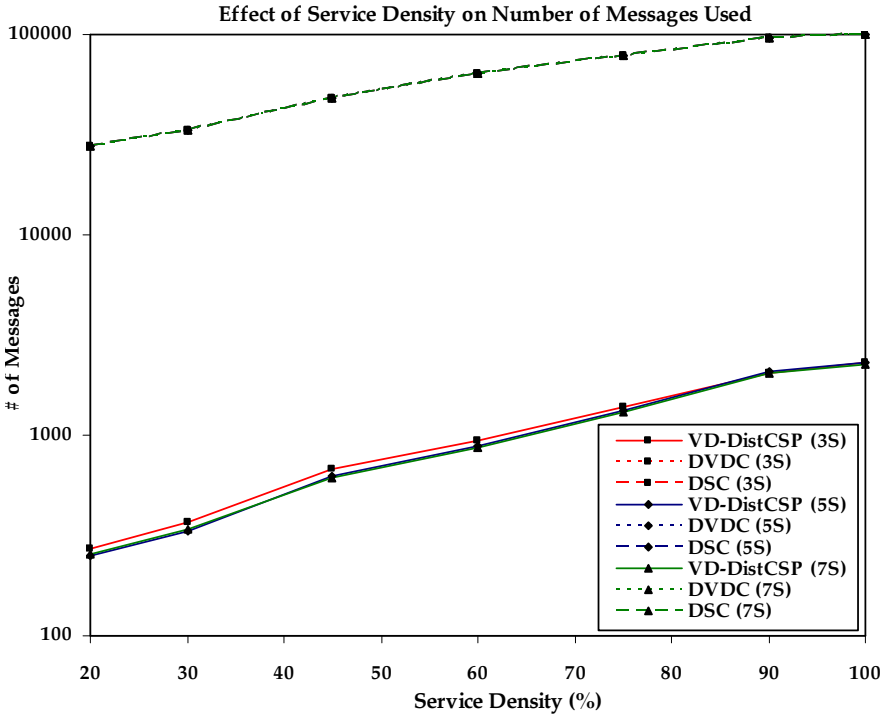


Fig. 6.6. Number of messages consumed with respect to service density for composition lengths 3, 5, 7.

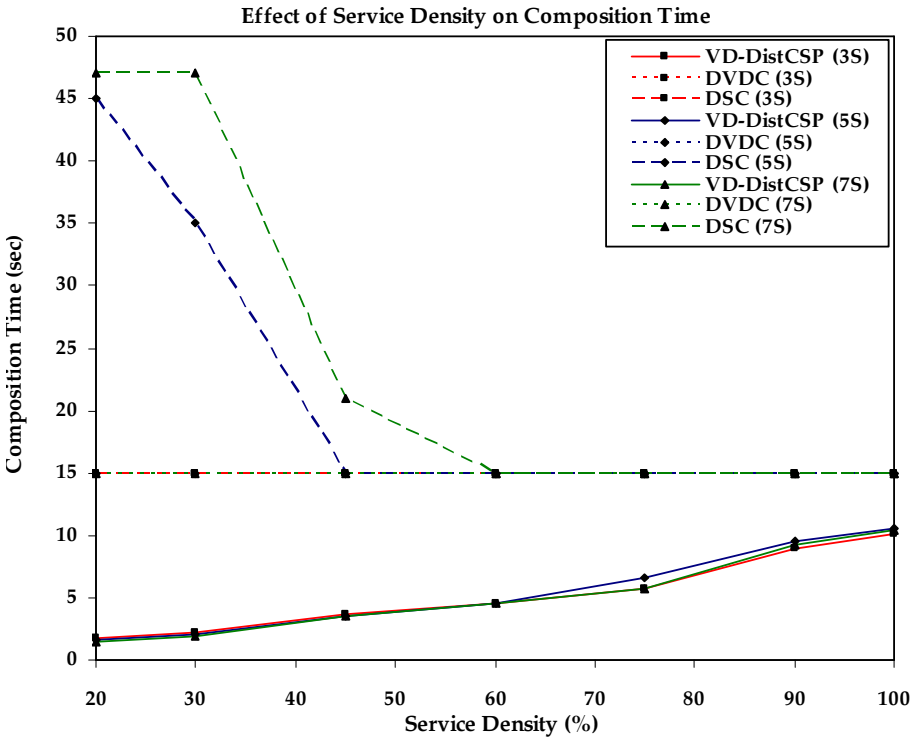


Fig. 6.7. Composition time with respect to service density for composition lengths 3, 5, 7.

6.5.4 Effects of Scaling

We further investigate the VD-DistCSP, DVDV, and DSC protocols by examining the effect scaling has on their performance. To achieve this end, we hold the service density at 50% and composition length at 3 services. Mobility continues to follow a random-way-point model with a minimum speed of 1m/s, a maximum speed of 3m/s, and a 5s stoppage time. Moreover, all broadcasts remain at a strict bounded hop count of 1 and the DVDC and DSC service advertising intervals are set to 5s. However, we now gradually increase the dimension area D of the experiment as well as the number of nodes N involved, such $D = 30 + 6 \times (\sqrt{N} - 5)$, while holding a nodes transmission range to 30m.

Results show that all three protocols suffer from scalability issues. The effect of scaling on the VD-DistCSP protocol proves to be detrimental to the achieved level of

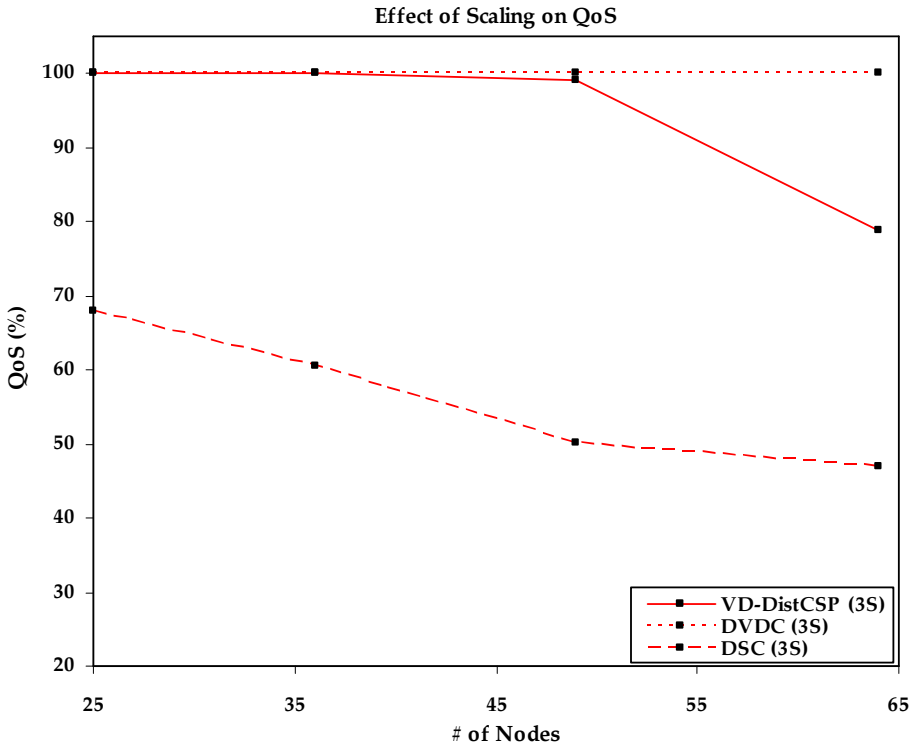


Fig. 6.8. QoS with respect to scaling number of nodes and dimension.

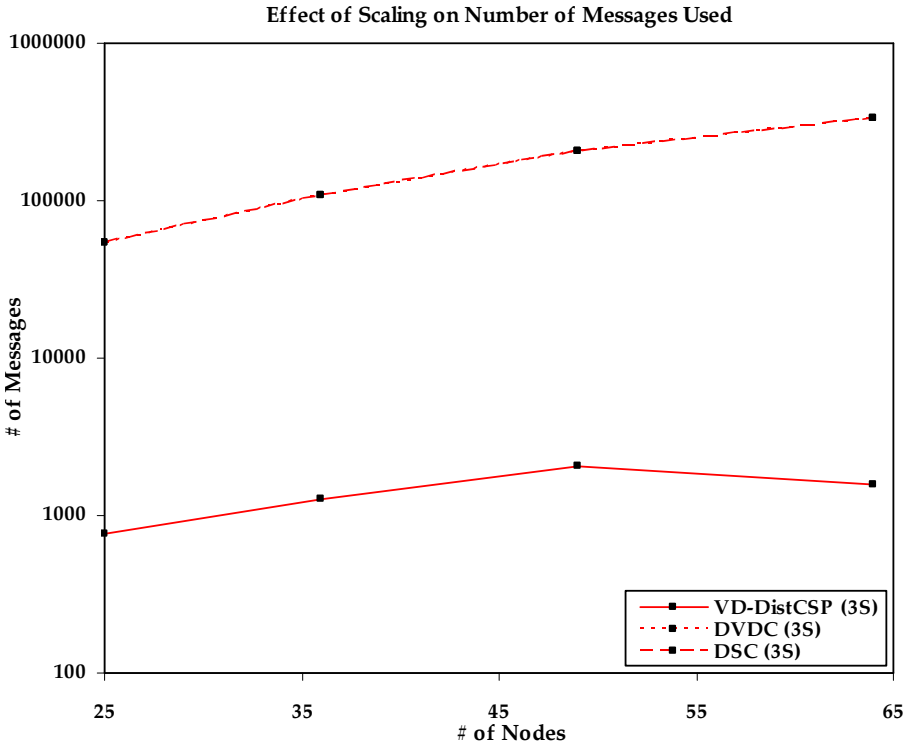


Fig. 6.9. Number of messages with respect to scaling number of nodes and dimension.

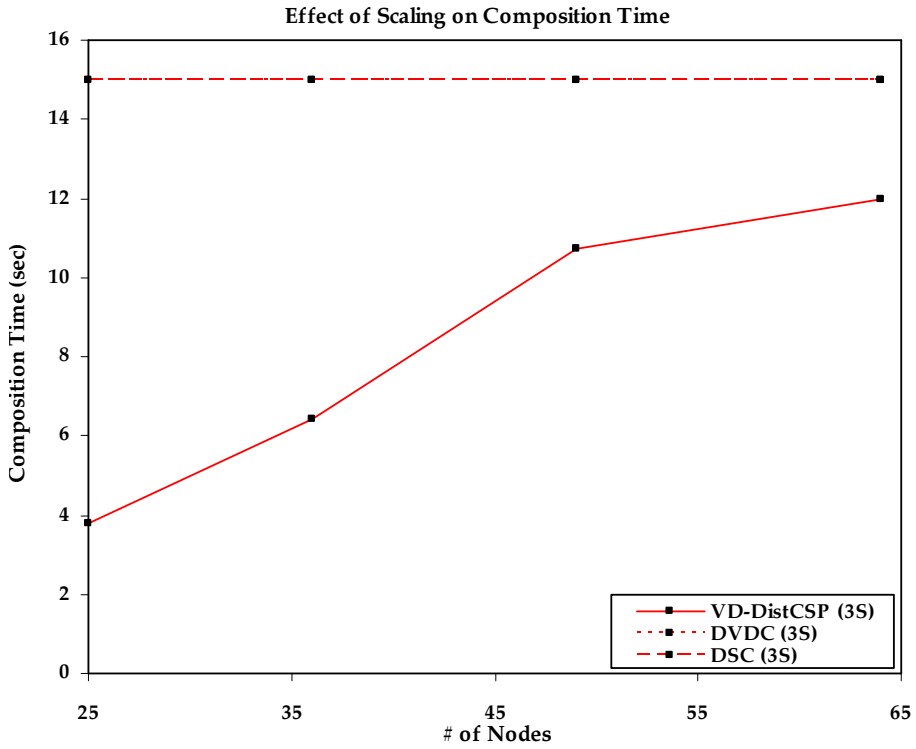


Fig. 6.10. Composition time with respect to scaling number of nodes and dimension.

QoS (Figure 6.8), such that as the dimension area surpasses the transmission range of a node, the performance begins a linear decrease. This can be attributed to both the lack of discovery of potential services by bounded broadcast and node disconnect during the negotiation process solving the CSP. This also leads to an increase in composition time (Figure 6.10), but only a small increase in messages (Figure 6.9), which taper down as the dimension continues to grow. The later can be attributed to the higher likelihood of more nodes being out of range of a node wishing to compose a service as the dimension increases, causing negotiations between fewer nodes. Unlike VD-DistCSP, the performance of DVDC remains at 100% in terms of QoS (Figure 6.8). This is due to the continuous broadcasting and heavy discovery mechanism embedded in the protocol. However, the cost in terms of messages required places a far too heavy burden on the network and is impractical. As expected, using the same discovery mechanism, DSC follows the same cost pattern as DVDC (Figure 6.9 and 6.10), but again only manages to

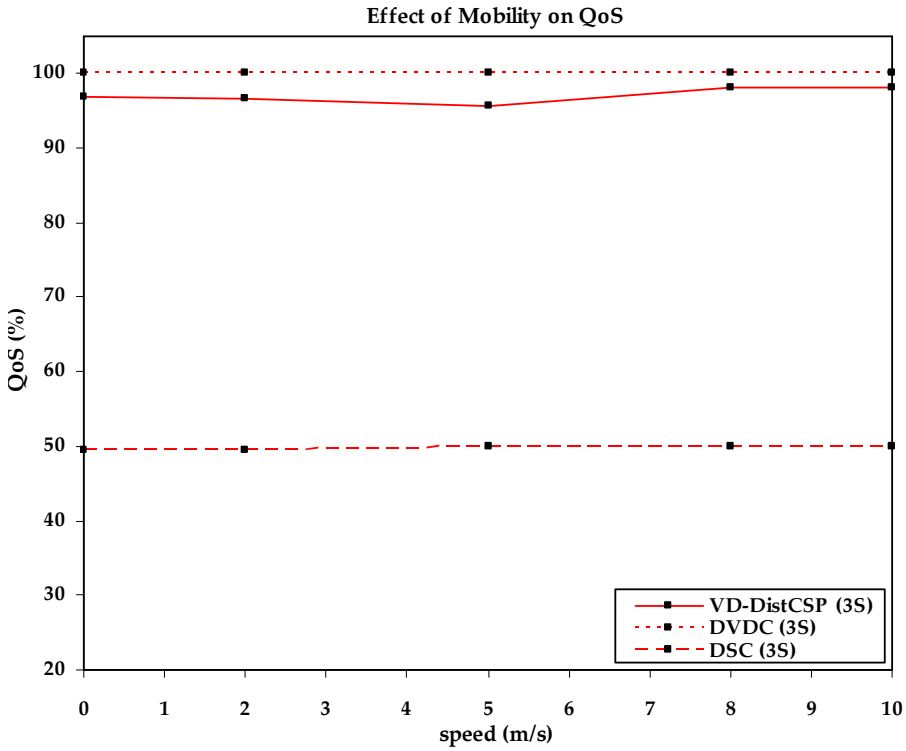


Fig. 6.11. QoS with respect to mobility speed.

acquire a QoS level following the average of the network (Figure 6.8), which decreases as the number of nodes increase.

6.5.5 Effects of Mobility

We now examine the effects of mobility on the VD-DistCSP, DVDV, and DSC protocols. To achieve this end, we hold the service density at 50%, composition length at 3 services, number of nodes at 49, dimension at 42x42m, and transmission range at 30m. Moreover, all broadcasts remain at a strict bounded hop count of 1 and the DVDC and DSC service advertising intervals are set to 5s. Mobility continues to follow a random-way-point model, but now gradually increases, with no stoppage time, from 0m/s to 10m/s.

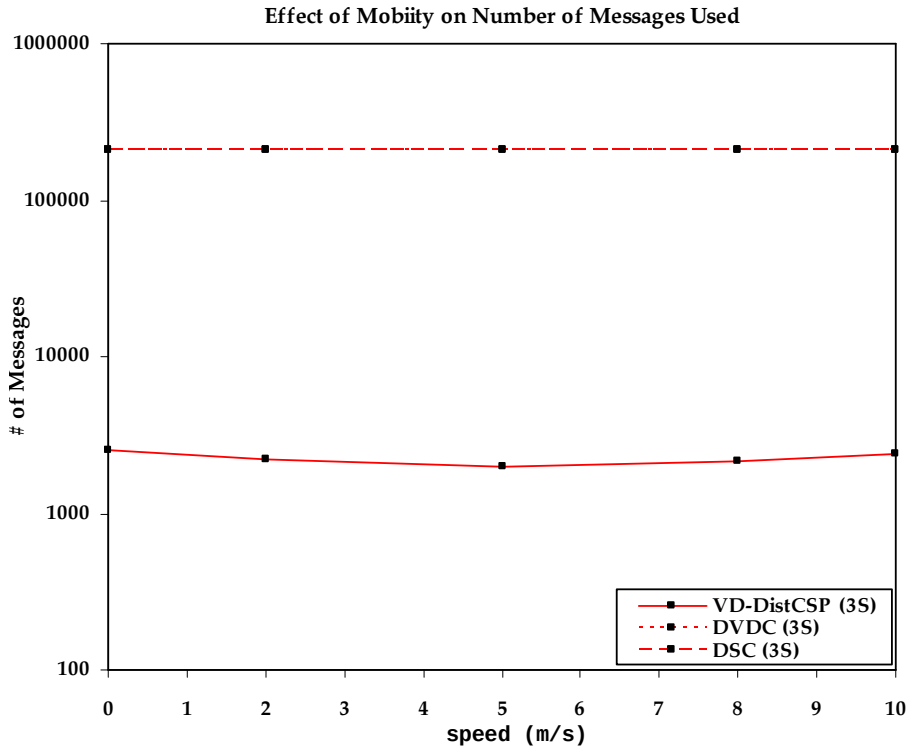


Fig. 6.12. Number of messages with respect to mobility speed.

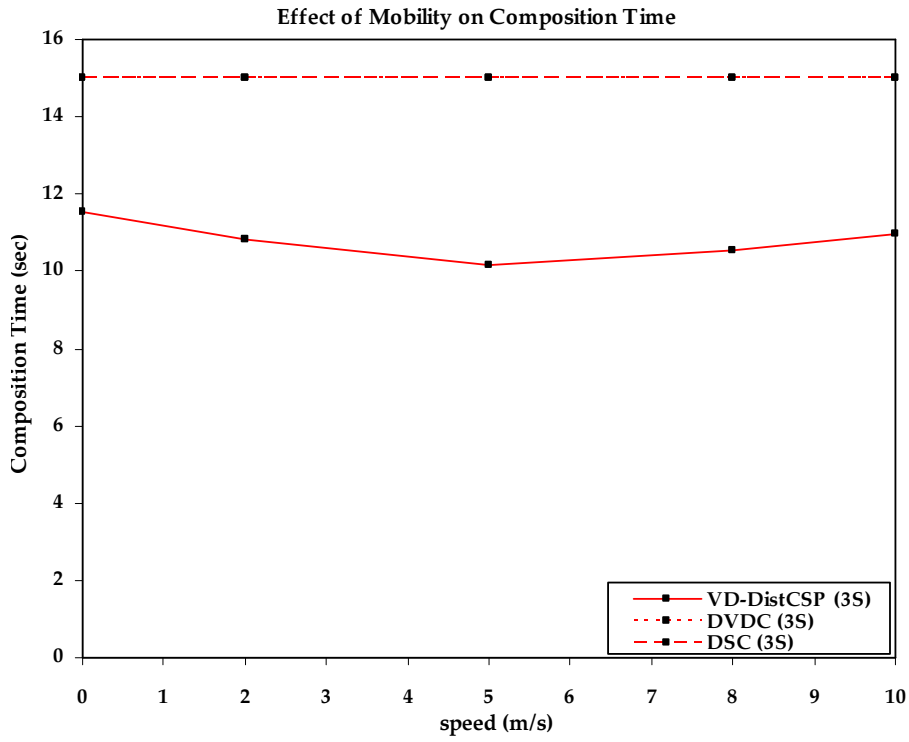


Fig. 6.13. Composition time with respect to mobility speed.

Results show that DSC's and DVDC's broadcast-based discovery mechanism is resilient to mobility, such that DVDC continuously discovers the best possible services for composition (Figure 6.11). Unlike DVDC, VD-DistCSP's pull-based service discovery mechanism suffers as mobility causes an increase in the number of nodes that are outside the range of the initiating node's bounded broadcast. This results in fewer nodes participating in the DistCSP and hence less time in solving the DistCSP, which is represented in both Figure 6.12 and 6.13. An interesting observation occurs when nodes reach a speed whereby, they move in and out of the initiating nodes range at a quick enough rate to be discovered and participate in the DistCSP. Hence, the subsequent rise in QoS, composition time and number of messages used. Although DVDC outperforms DistCSP in this experiment, the cost far outweighs the benefit, such that once again the number of messages required places a far too heavy burden on the network and is impractical.

In regards to DSC, the QoS remains as good as the average in the network, which remains steady (Figure 6.11).

6.6 Summary

In this Chapter, we have presented a distCSP model for task-based, QoS aware, virtual device composition in MANETs, and an asynchronous backtracking-based algorithm for solving the respective distCSP. In doing so, we have displayed a means of performing virtual device composition in MANETs without the use of continuous broadcast-based service advertisements, and without the need for agents to directly divulge neither complete nor precise information about their domain and constraints. Through simulation, we have shown the effectiveness of our method in achieving high QoS compositions without the unnecessary depletion of node resources.

However, lacking from the composition method is a means of reconciliation with respect to resolving capability differences (e.g. supported codecs, resolutions, bandwidth requirements, etc.) of discovered devices and the services they provide, potentially requiring the splitting and/or joining of media. In the following chapter, we address the issue of capability reconciliation for virtual device composition.

Chapter 7

Capability Reconciliation for Virtual Device composition

The previous Chapter provided a distCSP approach to virtual device composition without the need for continuous broadcast-based service advertisements, and without the need for agents to directly divulge neither complete nor precise information about their domain and constraints. However, the dynamic and ad hoc nature of the discovery and composition of such devices and the services they provide inevitably leads to capability differences, whereby the input of a service B is not compatible with the output of a service A; A and B needing to be composed. In this Chapter, we present a distributed constraint satisfaction problem for capability reconciliation in MANETs, and through simulation show its effectiveness and efficiency.

7.1 Introduction

In Chapters 4-6 we addressed the dynamic composition and management of virtual devices for ad hoc multimedia service delivery, where we introduced a novel SON-based definition of a virtual device applied to an autonomous policy driven framework for virtual device management formed as a hierarchical structure of distributed elements, including autonomic elements, all working towards the self-management of virtual

devices. Moreover, we developed for the framework in Chapter 6 a distributed constraint satisfaction problem (distCSP) model for task-based, quality-of-service aware, virtual device composition in MANETs, and a virtual device distCSP protocol utilizing an asynchronous backtracking-based algorithm for solving the respective distCSP. The benefit of using a distCSP approach is the ability to provide composition without the need for continuous broadcast-based service advertisements, and without the need for agents to directly divulge neither complete nor precise information about their domain and constraints; two factors ailing most MANET-based composition techniques.

However, lacking from the composition method is a means of reconciliation with respect to resolving capability differences (e.g. supported codecs, resolutions, bandwidth requirements, etc.) of discovered devices and the services they provide, potentially requiring the splitting and/or joining of media. In this Chapter we examine the possibility of extending our distributed constraint satisfaction problem (distCSP) model to include a method for capability reconciliation. The resulting contributions of this Chapter are a distCSP model for capability reconciliation in MANETs and a respective protocol utilizing an asynchronous backtracking-based algorithm for solving the particular distCSP. Through simulation and comparison we show the effectiveness of our method.

The rest of this Chapter is organized as follows. Section 7.2 and 7.3 present the problem formulation and model, respectively. Section 7.4 describes our protocol and asynchronous backtracking-based algorithm. Section 7.5 presents our simulation results, and Section 7.6 gives our conclusions.

7.2 Problem Formulation

We once again consider a mobile ad hoc environment composed of mobile nodes connected with one another by ad hoc network protocols. Represented by some nodes are networked devices providing single or multiple services, whereby peer nodes are able to invoke these services. A virtual device occurs as services provided by devices in a user’s vicinity compose in satisfying an atomic task. We define an *atomic task* as an application level generated task, independent from any other task, requiring one or more

atomic services. We define an *atomic service* as a service residing on a single node, which can consist of further components if and only if they too reside on the same single node.

Produced from a task is a *task-based composition specification (TCspec)*, which we define as a specification of service and service property requirements, represented as policies (shown in Figure 6.1). Policies are classified into three categories: task requirement, service constraint, and user preference. *Task requirements* represent the minimum services required for task satisfaction. *Service constraints* represent the necessary constraints on the services to allow the composed service to function as the application intended. *User preferences* are service constraints controlled by the user, which are present solely to shape the user's desired quality of service (QoS). As numerous composition permutations exist in the environment, we refer to QoS, such that separating one composition from another is their respective levels of quality in terms of service and user experience. We seek to maximize the QoS generated by a composed virtual device, by efficiently probing possible composition permutations in a user's vicinity and identifying the configuration that satisfies all service requirements, adheres to all service constraints, and best caterers to the users' identified preferences.

In Chapter 6 we provided a method for achieving this end, based on modeling and solving virtual device composition as a distributed constraint satisfaction problem (distCSP). However, missing from this work, is a means of reconciliation with respect to resolving capability differences (e.g. supported codecs, resolutions, bandwidth requirements, etc.) of discovered devices and the services they provide, potentially requiring the splitting and/or joining of media. In the following we revisit the problem of capability reconciliation from a perspective fitting our architecture presented in Chapter 4.

7.2.1 Capability Reconciliation

We define services as falling into one of three categories (forgoing Media Controllers for simplicity): Media Service Server (MSS), Media Service Port (MSP), and Media

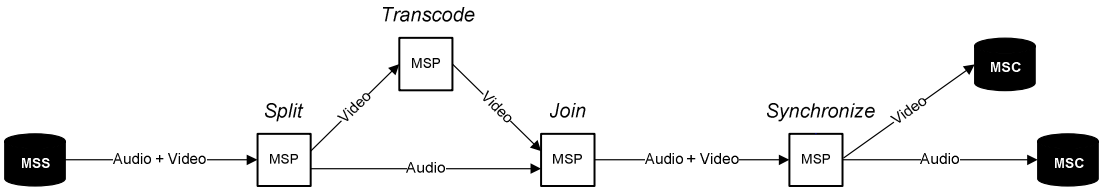
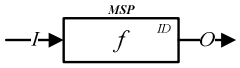


Fig. 7.1. A VDMA-based sample capability reconciliation.

Service Client (MSC). *Media service clients* are services that consume media (e.g. flat screen, speaker, etc.), as opposed to *media service servers*, which are services that provide media for consumption (e.g. media server). *Media service ports* are services providing media processing (e.g. media transformation, splitting, caching, routing, etc.), utilized, as media flows from media servers to clients. Our current methods successfully discover and compose a virtual device made up of MSS and MSCs for composition, but do not provide a mean for identifying and discovering appropriate MSPs allowing media to flow from MSS to MSCs in the presence of capability differences. The problem can formally be defined as follows:



An MSP can be regarded as a function *f* applied on an input *I* to produce output *O*, whereby an MSP service can be described using a service identification *ID*. Each MSP service consumed incurs a cost and each MSP provides one or more services. As MSCs and MSSs do not alter media flow they can be described using respective *I* and *O* variables. The input I of an MSC refers to a possible input format, while the output O refers to the content output channel. The input I of an MSS refers to content input and the output O refers to an encoding scheme. An MSC can be directly serviced by a MSS if the input I of the MSC in compatible to the output O of the MSS. Should this not be the case, one or more MSPs are required to be inserted between the MSC and MSS in order to establish successful media delivery. Actual media processing services can be decomposed into the composition of four atomic classes (i.e. filter, transformer, multiplexer, and demultiplexer), connected in serial or parallel [45]. Each of these classes serve a particular function and are defined in Chapter 2.3.2.

Provided in Figure 7.1 is a sample media stream being adapted from an MSS to an MSC. The adaptation sequence depicts a scenario whereby a single media stream containing audio and video is sent from the MSS. In order for the MSC to correctly view the media, the video codec must be adapted to one that it supports. Therefore, the stream is first split into its component parts, at which point the video can be transcoded, rejoined with the audio, and then buffered for consumption by the MSC.

7.3 Problem Modeling

In solving the capability reconciliation problem, it is first necessary to identify the necessary logical transformations between media endpoints (i.e MSS and MSC). Moreover, it is also necessary to determine the effect an MSP will have on the media content flow. We assume the existence of a function $Sim(MSS, MSC_{1...z})$ able to compute the difference between media endpoints. $Sim(MSS, MSC_{1...z}) = Ø$ depicts the input of the MSC as compatible with the output of MSS, whereas $Sim(MSS, MSC_{1...z}) = \lambda$ represents the set λ of k required adaptations $\lambda[0]$ to $\lambda[k]$ for the media to achieve compatibility as it flows from MSS to MSC(s).

The addition of MSPs in achieving media compatibility inevitably comes at a cost, such that these services may increase delay, decrease throughput, and consume network resources (e.g. battery power, processing power/load). Furthermore, the use of the service, potentially included as part of service provider agreement, has a monetary cost to the user. Therefore, the composed MSP service path allowing for the media flow should minimize a cost criterion, described bellow, including consideration for both monetary and network cost. We do not develop nor impose strict parameters on the criterion as it is application and user dependent.

In satisfying the outlined objective, we model the capability reconciliation problem in the form of a distributed constraint satisfaction problem $RC\text{-}DistCSP = (X, D, C)$ where:

- $X = \{X_i\}(i = 1,..., n; X_i \subseteq D)$. X_i is a variable corresponding to the set of media processing functions being provided by node N_i to satisfy a particular media flow λ.

- $D = \{D_i\}(i = \{1,...,n\})$. D is a set of n domains of values, such that D_i is the domain of media processing functions capable of being provided by X_i.
- C is the set of constraints of the problem. We can formulate them as follows:

$$\forall_{i,\,j} \quad X_i \cap X_j = \{\ \}\ \ i, j \in \{1,...,n\}, i \neq j \tag{1}$$

$$\forall_i \quad \bigcup X_i = \lambda, \text{ where } \lambda = Sim(MSS, MSC_{1...z}) \tag{2}$$

$$\textit{Select } N_i \text{ for } \lambda\,[m] \text{ such that } \lambda\,[m] \in D_i \text{ and} \tag{3}$$

$$Min(W_{mcf}\, MCF(N_i) + W_{ncf}\, NCF(N_{j,\,\lambda\,[m-1]}, N_{i,\,\lambda\,[m]})$$

$$\beta = \sum_{m=0}^{k} MCF(N\lambda[m]) + NCF(N\lambda[m-1], N\lambda[m]) \tag{4}$$

$$\text{Min}(\beta)$$

λ is a set of media adaptations corresponding to all k required adaptations $\lambda[m]$, $m = \{0,\ldots,k\}$. Constraint (1) ensures that at most one node provides any particular media processing function. Constraint (2) guarantees that all required adaptations have been assigned. Constraint (3) requires that the node selected for the provision of a particular media processing functionality is 1) that minimizing the monetary cost of the composed media flow where $MCF(N_i)$ represents the *monetary cost function* (MCF) of a particular node N_i, and 2) that minimizing the network cost of the composed media flow, where $NCF(N_{j,\,\lambda\,[m-1]}, N_{i,\,\lambda\,[m]})$ represents the *network cost function* (NCF) incurred from media adaptation $\lambda\ [m-1]$ occurring and flowing from node N_j to node N_i providing media adaptation $\lambda\ [m]$. Constraint (3) is influenced by a pair of weights (i.e. W_{mcf} and W_{ncf}) identifying whether priority should be given to monetary or network cost. Such weights can be set based on agreements between the consumer, service provider, and network administrators. Constraint (4) represents the overall cost of capability reconciliation β of the final composition path, whereby the path chosen should be that minimizing β.

Having identified our constraint satisfaction problem, we apply an algorithm based on the asynchronous backtracking algorithm [116] to solve the capability reconciliation problem in MANETs. We call our method the *capability reconciliation DistCSP* (CRD) protocol and our respective algorithm the *capability reconciliation backtracking* (CR-Btracking) algorithm.

7.4 Capability Reconciliation DistCSP Protocol

The capability reconciliation distCSP (CRD) protocol allows automated agents, each representing the interests of a particular node, to act concurrently and asynchronously without any global control, in deciding how to form the best possible media flow in providing the necessary media processing functionality.

The capability reconciliation problem, from a distCSP perspective, can be represented as an $m \times n$ Matrix M (Figure 7.2.i), comprised of m nodes N_0, N_1, … , N_m, and n necessary adaptations λ[0], λ[1], … , λ[n] representing the media flow options in reconciliation such that:

- A black circle in $M_{x,y}$ represents an adaptation λ[y], where N_x provides adaptation λ[y].

The solution to the virtual device composition problem, requires the application of the constraint set C defined in Section 7.3, whereby a Matrix $M'_{m,n}$ (Figure 7.2.ii) is formed such that:

- A black circle in $M_{x,y}$ containing the letter p represents a *potential* adaptation λ[y] in a node N_x's X_x variable, such that N_x has the potential of contributing adaptation λ[y] in the capability reconciliation.
- A black circle in $M_{x,y}$ containing the letter c represents a *commited* adaptation λ[y] in a node N_x's X_x variable, such that N_x has committed to contributing adaptation λ[y] in the capability reconciliation.
- Each column n must contain *exactly one* black circle containing the letter c.
- The composition of node/adaptation pairs corresponding to the black circle values containing the letter c form a capability reconciliation satisfying the constraint set C.

The CRD protocol establishes a Matrix M and provides a concurrent and asynchronous negotiation mechanism to transform Matrix M into Matrix M'. The following presents the various steps of the CRD protocol.

7.4.1 Candidate Formation

The CRD protocol begins with the process of candidate formation in constructing the

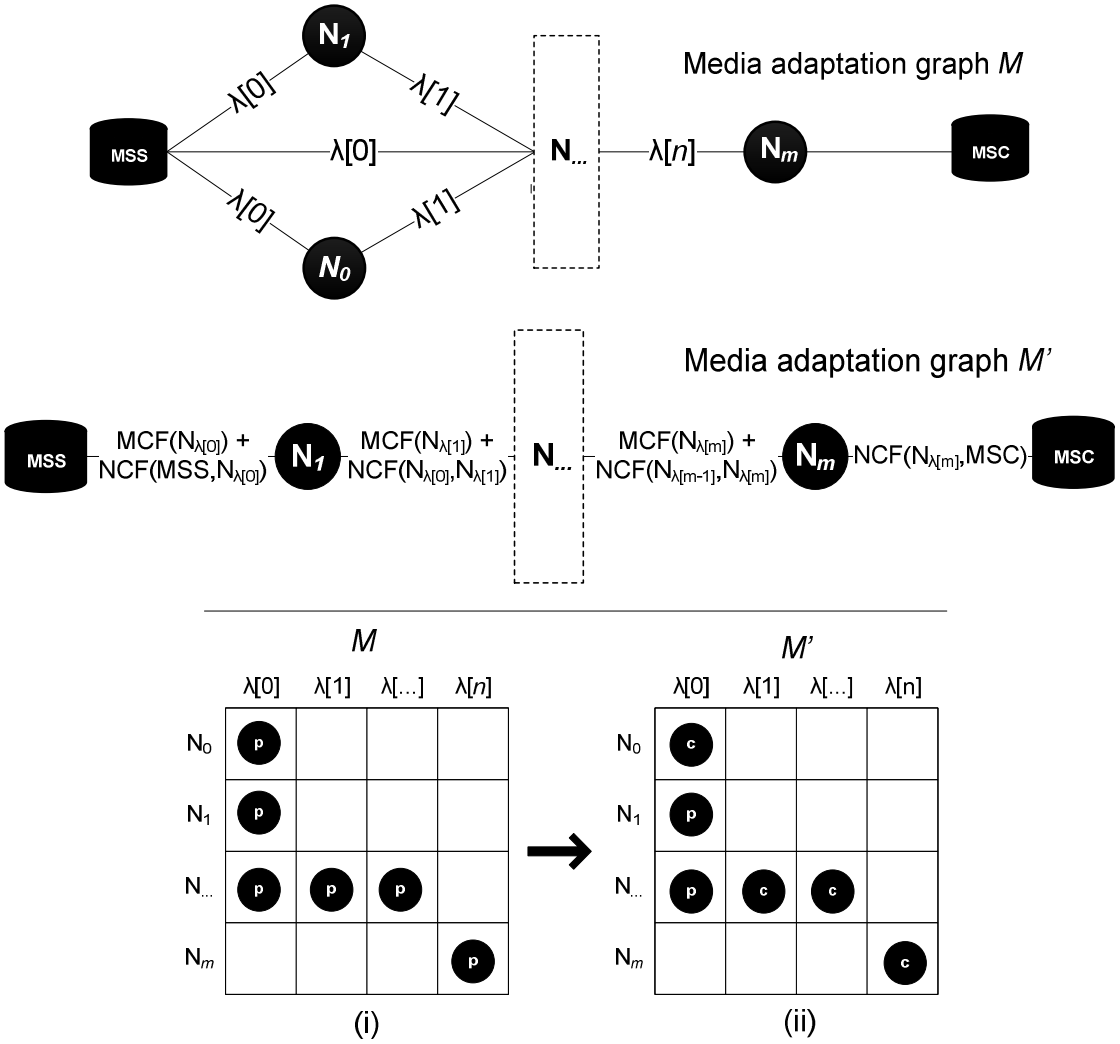


Fig. 7.2. A transformation from Matrix M to M', visually abstracting the capability reconciliation problem accompanied by respective media flow adaptation graphs.

rows of Matrix M, which initiates execution after the discovery of respective MSC(s).Utilizing the similarity function previously defined, the set of adaptations λ are identified and embedded in a capability reconciliation request (*CRrequest*) message which is sent out into the user's vicinity, using a bounded broadcast by the node initially requesting the composition of a virtual device. A node receiving the *CRrequest* and providing one or more of the MSP adaptation services in λ responds with the sending of a capability reconciliation reply (*CRreply*) message. The CRrequest initiator utilizes these replies to form a *candidate table* containing the addresses and unique identifiers of all responding nodes. The nodes in the candidate table are organized in *descending*

priority order based on their unique identifiers. Once the table is formed, the *CRrequest* initiator distributes the candidate table to all respective candidates.

7.4.2 CR-Btracking Algorithm

Now that Matrix *M* is complete, we apply the capability reconciliation backtracking (CR-Btracking) algorithm, which is based on the traditional asynchronous bactracking algorithm for solving distCSPs [116]. The execution of the CR-Btracking algorithm (partial pseudocode provided in Figure 7.3) is asynchronous between nodes in matrix *M* and begins with all candidate nodes with potential for service λ[0] computing the sum of their respective $W_{mcf}\ MCF(N_{\lambda[0]}) + W_{ncf}\ NCF(MSS, N_{\lambda[0]})$ value, which they place in a variable $\beta_{current}$, and placing their identifier in a variable *comPath*. $\beta_{current}$ represents the current overall cost of capability reconciliation for the specific composition path *comPath*. Moreover in a tertiary, the following values are placed: λ_{Ni}, $\beta_{current}$, and *comPath*. λ_{Ni} is the identifier of the specific adaptation service being contributed. For each of a node N_i's *j* adaptation contributions a tertiary is stored in N_i's X_i, such that $X_i = \{(\lambda[m], \beta_{current}, comPath)\}(m \in \{1,\ldots,k\})$.

Similarly to VD-Btracking (Section 6.4.3), upon forming their tertiary, nodes with potential for service λ[0] send these values to all nodes in the candidate table using *ok?* messages. Following the receipt of an *ok?* message (Fig. 7.3.i), a node N_i compares its X_i with its agent view (Fig. 8.iii), looking for any commitment contributions as well as inconsistencies with regards to *higher priority nodes*. A commitment contribution occurs when a node N_i has received knowledge from a node N_j which has committed to adaptation service λ[m-1], where N_i has potential of contributing service λ[m]. In this case N_i computes its $W_{mcf}\ MCF(N_i) + W_{ncf}\ NCF(N_{j,\lambda[m-1]}, N_{i,\lambda[m]})$, value and adds this to $\beta_{current}$ provided by N_j, now forming a new tertiary altering its X_i where it also adds its identifier to *comPath*. This new X_i is subsequentlty distributed in an *ok?* message to all candidates.

Again, inconsistencies occur when N_i's X_i value violates the set of constraints *C* of the problem. If inconsistencies are present, N_i attempts to alter its X_i to resolve the inconsistencies. If N_i is able to achieve consistency, it distributes its new X_i in an *ok?*

```
when received (ok?, [N_i,X_i]) do – (i)
  add[N_i,X_i] to agent_view;
  check_agent_view;
end do;

when received (nogood, N_i, nogood) do – (ii)
  if provide λ[m] and nogood > my β_current
    then ignore;
  else
    add nogood to nogood_list;
  end if;
  check_agent_view;
end do;

procedure check_agent_view – (iii)
  when it is possible to contribute do
    goContribute
    send (ok?,(N_self, X_self) to candidates;
  end do;
  if provide λ[m] then
    findMinβ
    purgePathsExceptMinβ
    send (ok?,(N_self, X_self) to candidates;
  end if;
  when agent_view and X_self are not consistent do
   if no value in D_self is consistent with
   agent_view then backtrack;
   else
     select d∈D_self where agent_view and d        are
     consistent;
     X_self ← d;
     send (ok?,(N_self, X_self) to candidates;
    end if;
  end do;

procedure backtrack – (iv)
  nogoods ← {V|V = inconsistent subset of agent_view};
  when an empty set is an element of nogoods do
    broadcast to all candidates that there is no solution,
    terminate this algorithm;
  end do;
  for each V ∈ nogoods do
     send (nogood, N_self, V) to N_j∈V
  end do;
```

Fig. 7.3. Partial CR-Btracking algorithm pseudocode.

message to all candidates. If no such X_i exists, then it is necessary to *backtrack* (Fig. 7.3.iv). Backtracking involves the sending of one or more *nogood* messages to nodes causing the inconsistency. A *nogood* message states to the receiver exactly what tertiary value in X_i to deem as *nogood*. Every candidate maintains a *nogood list*, which

contributes to its constraint set *C*. Upon receiving a *nogood* message, the recipient attempts to manipulate its respective X_i value, allowing the sender to re-enter a consistent state (Fig. 7.3.ii). If the *nogood* recipient is unable to resolve the inconsistency, then it too triggers a backtrack. A no solution presents itself when the highest priority node cannot resolve a *nogood*, such that it has exhausted all of its variable options. This results in termination of the algorithm by failure as a consequence of the problem being over constrained.

In the CR-Btracking algorithm, inconsistencies do not occur until knowledge reaches candidate nodes in the form of *ok?* messages containing a complete path from MSS to MSC. It is the responsibility of a node N_i providing the final adaptation service $\lambda[k]$ in λ to:

1. Compute the final composition by forming the final tertiary containing the total β as well as complete composition path, which includes the computation of $NCF(N_{\lambda[k]}, MSC)$.
2. Examine the final tertiary values whereby it provides the final composition, identifying the composition minimizing $\beta_{current}$, removing all other tertiaries from its X_i , and sending *nogood* messages to all nodes in the *comPaths* not minimizing $\beta_{current}$ specifying that they remove the tertiary coupled to the specific *comPath* from their own X_i.
3. Examine the final tertiary values of *higher priority nodes* providing the final adaptation in column $\lambda[k]$, such that should it find a $\beta_{current}$ smaller than its own it will remove its specific tertiary from its X_i classifying it as no good and then proceed to send a *nogood* message to all nodes identified in the *comPath* specifying that they remove the tertiary coupled to the specific *comPath* from their own X_i. On the other hand, should it find a $\beta_{current}$ larger than its own, it will send a no good to the respective node specifying that node's *comPath* as *nogood*. That node responds by removing the specific tertiary and sending *nogoods* to the remaining nodes in the path.

A solution presents itself when all nodes share a common agent view and are in a consistent state. In this case only one composition path for each MSC should remain in every agent view; hence the final solution. Nodes continue to periodically send *ok?*

messages until either a no solution or final solution is reached. With regards to complexity, constraint satisfaction is generally considered NP-complete [116]. As a result, worst-case time complexity is exponential in the number of variables *n*, $X = \{X_i\}(i = 1,...,n; X_i \subseteq D)$.

7.4.3 Example Execution

In clarifying the algorithm we provide a sample execution. It should be noted that the trace of the algorithm's execution can vary according to timing/delay of messages, and that this example shows one possible trace of execution.

The initial values are shown in Figure 7.4.a, whereby candidate formation has taken place and priority has been determined by numerical identifiers. For the benefit of visual clarification, in this example we assume NCF(*MSS*, $N_{\lambda[0]}$) and *NCF*($N_{\lambda[k]}$, *MSC*) to be negligible. Values within the black circle identitfy a particular node's monetary cost, whereby values placed on links between black circles identify the network cost between the two connected nodes.

Node's able to provide λ[0] begin by computing $\beta_{current}$, and *comPath*, which they distribute using *ok?* messages (Figure 7.4.b). N_2 receives *ok?* messages from N_0, N_1, and N_3 computing and adding itself to the respective $\beta_{currents}$ and *comPaths*, and begins spreading this knowledge in *ok?* messages as well (Figure 7.4.c).

N_3 receives N_2's *ok?* message and goes through the same procedure as N_2 (Figure 7.4.d). When N_2 and N_0 receive N_3's *ok?*message (Figure 7.4.e) they both compute their respective final $\beta_{currents}$ *and* comPaths, identify the comPath minimizing $\beta_{current}$ and purge all tertiaries not minimizing $\beta_{current}$. In this case N_3 finds that N_0 provides a $\beta_{current}$ smaller than its minimum, and therefore completely purges all of its tertiaries, sending appropriate *nogoods* (Figure 7.4.f). N_0 completes the process by sending *nogoods* to nodes in the *comPath* not minimizing $\beta_{current}$ (Figure 7.4.h). At this point the only tertiaries remaining in every node's X_i are those forming the final solution (Figure 7.4.i).

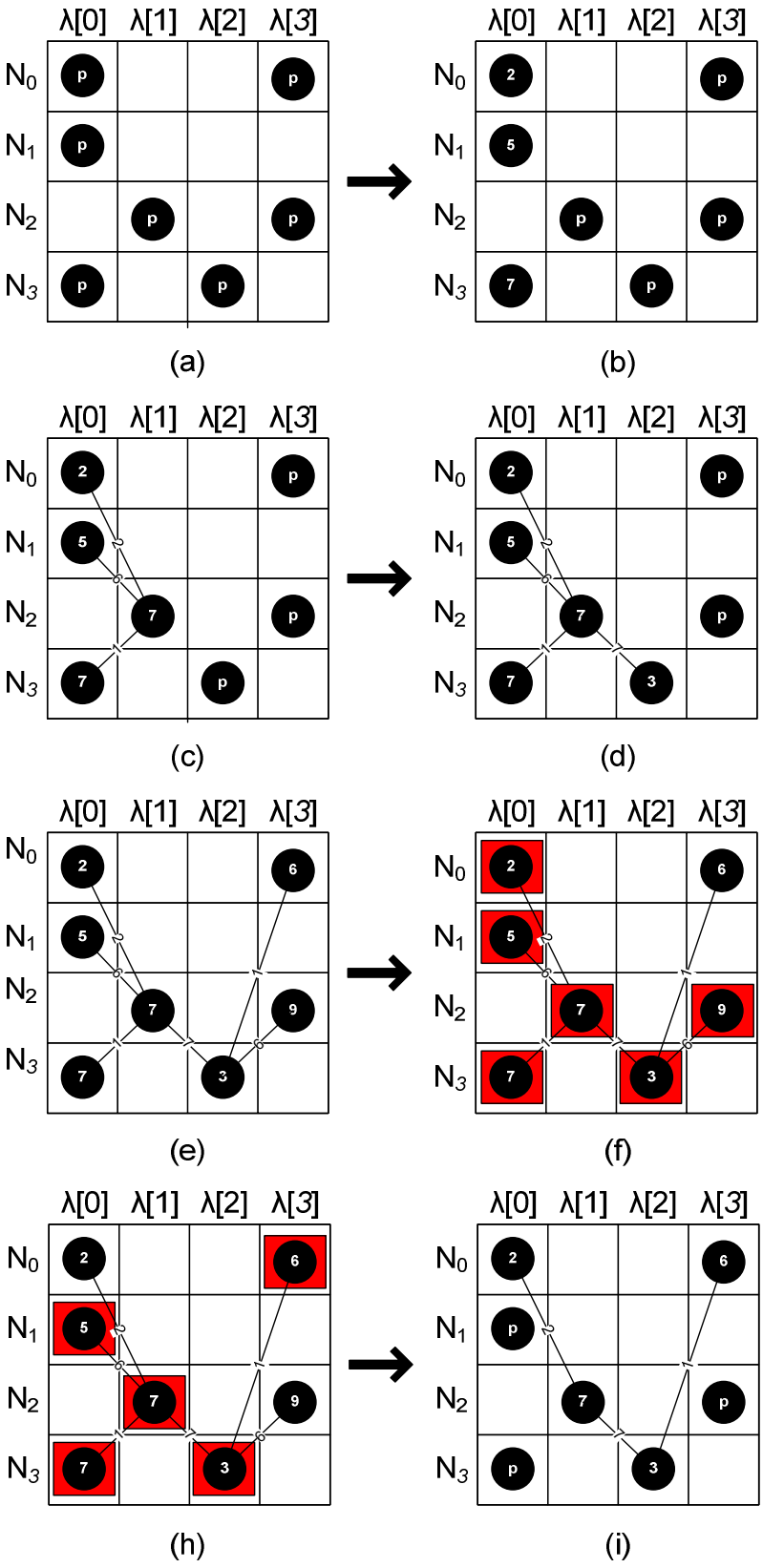


Fig. 7.4. Example execution of the CR-Btracking algorithm.

7.5 Simulation Results and Analysis

This section evaluates the performance of our CRD protocol for different composite lengths in varying service densities, mobility, and topologies. We compare our results to those of D. Chakraborty *et. al.*'s Distributed Service Composition protocol (DSC) [65], discussed in Section 2, in an effort to identify performance and efficiency gain in terms of message usage and composition time.

7.5.1 Simulation Parameters and Metrics

We build a MANET and implement the CRD and DSC protocols using J-SIM [120], a well known simulation environment. In an attempt to simulate a home environment, we employ an area of 30x30m containing 25 nodes with a transmission range of 30m. All nodes follow a random-way-point mobility model with a minimum speed of 1m/s, a maximum speed of 3m/s, and a 5s stoppage time. All broadcasts follow a strict bounded hop count of 1 and DSCs service advertising intervals are set to 5s. Simulations imitate a

Duration	100 requests over 24 hour period
Space(x,y)	30×30 m
Transmission range	30 m
No. of nodes	25
Advertisement interval (DVDC and DSC)	5 sec
Advertisement lifetime (DVDC and DSC)	7 sec
Control hop count	1
Mobility	Random-way-point model with a minimum speed of 1m/s, a maximum speed of 3m/s, and a 5s stoppage time
Initial topology	Grid topology with nodes equally spaced out in (x,y)
Composite lengths	3, 5, and 7
Service density	20 to 100%

Table 7.1. Simulation parameter summary.

mobile device entering a home environment and making 100 requests over a 24 hour period. The simulation is repeated for composite lengths 3, 5, and 7, and service densities 20-100%. Service density is defined as the percentage of nodes containing one of the adaptation services present in λ. Identified in each simulation is how closely the composed composition path matches the optimal path present in the network, and the amount of time and number of messages consumed in this process.

7.5.2 QoS Analysis

From Fig 7.5, results indicate that CRD is able to continuously discover the optimal composition path for capability reconciliation (i.e. lines overlap at 100%). Although DSC also does discover composition paths for capability reconciliation, on average they are only 58% optimal. This is due to the fact that no mechanism exists within DSC to find an optimal path, whereby DSC's performance follows the average β available in the network, which gradually increases as service density rises. It should be noted that lines in Fig 7.5. representing composition lengths 5 and 7, begin at 30% and 45% service density, respectively, such that insufficient adaptation services exist in the network to successfully reconcile the particular capability differences bellow those values.

7.5.3 Cost Analysis

Taken as an average per single composition, the number of messages used by CRD (Figure 7.6) is considerably lower than that of DSC, using an average of 98% fewer messages. Moreover, an average composition time decrease by CRD of 16.17 seconds (78%) is present when compared to DSC (Figure 7.7), such that DSC holds at about 15 seconds for all service densities shown in Figure 7.7. This significant drop in messages can be attributed to DSC's reliance on nodes to advertise their services at periodic intervals in order to ensure up-to-date service knowledge in the network. Furthermore, DSC is a broker based protocol requiring a process of broker election, service discovery, and service aggregation phases, which is time consuming, as opposed to CRD efficiently solving the problem as a single distCSP. In addition, the initial spike of DSC for

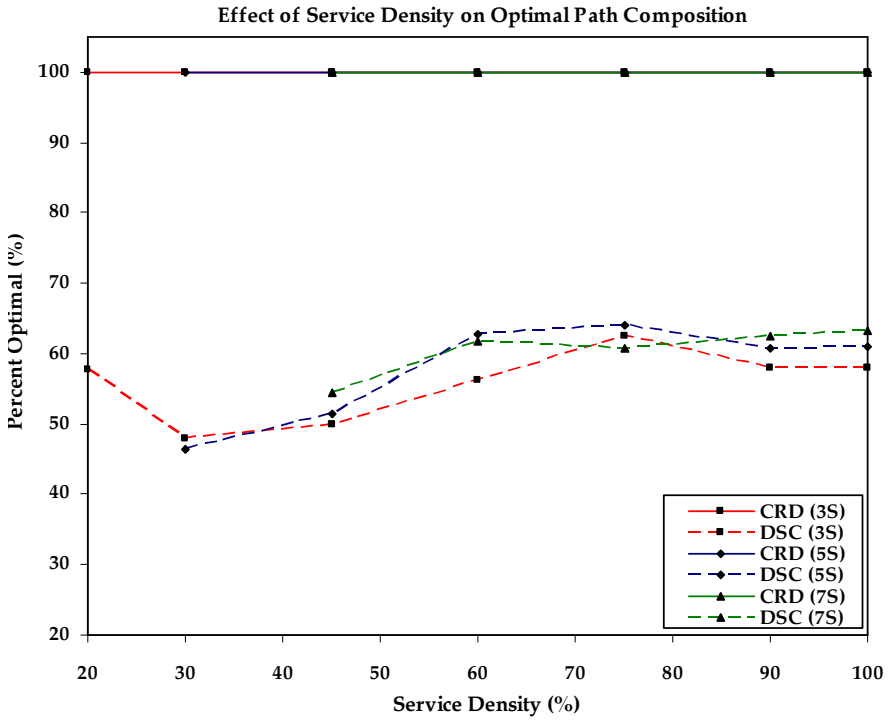


Fig. 7.5. Optimal Path Composition with respect to service density for composition lengths 3, 5, 7.

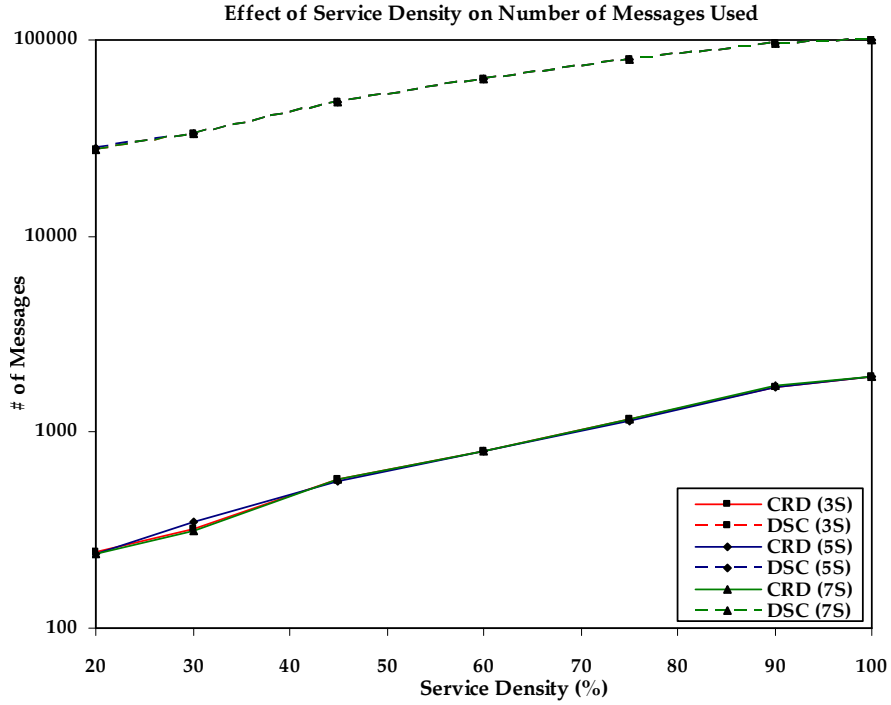


Fig. 7.6. Number of messages consumed with respect to service density for composition lengths 3, 5, 7.

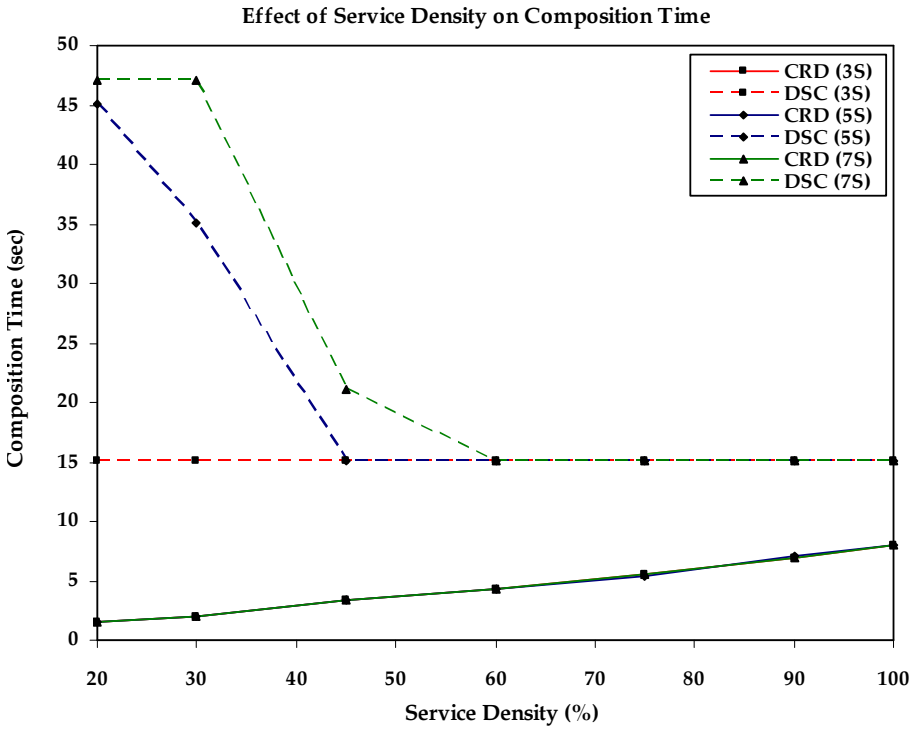


Fig. 7.7. Composition time with respect to service density for composition lengths 3, 5, 7.

composition lengths 5 and 7 in Figure 7.7, can be accredited to DSC making fruitless attempts to discover adaptations services that do not exist, whereby CRD, in finding the optimal composition path, is able to swiftly recognize a desired service as existing or not.

As CRD utilizes the same mechanisms for discovering services as the VD-DistSCP protocol, it produces results identical to those provided by VD-DistCSP in Chapter 6 with respect to scalability and mobility.

7.6 Summary

In this Chapter, we have presented a distCSP model for capability reconciliation in MANETs and a respective protocol utilizing an asynchronous backtracking-based algorithm for solving the particular distCSP. In doing so, we have defined and addressed an issue missing from current schemes as well as our own previous work in allowing for the automatic composition of devices, and the services they provide, to create dynamic

service configurations with limited human intervention in the presence of capability differences. Through simulation, we have shown the effectiveness of our method in achieving optimal composition paths for capability reconciliation without the unnecessary depletion of node resources.

Up to this point we have mainly considered the creation of the best possible virtual device at the time of need. In terms of post creation, assuming that a composed virtual device will remain valid in a static form is incorrect. As the network changes, the virtual device should adapt to these changes in a manner providing minimal disruption to the user and remain the best possible virtual device at the time of need. In the following Chapter, we address this issue.

Chapter 8

Automated Adaptation for Virtual Device Composition

The virtual device concept enables services provided by networked appliances in a user's vicinity to intelligently compose themselves for exploitation by the user. The application of this concept should function in highly dynamic environments occurring throughout a user's daily encounters. Assuming that a composed virtual device is to remain static in nature once generated is false, as the virtual device should gracefully degrade and upgrade along with the conditions available in the user's environment; particularly, the current performance requirements of the network. Current schemes for infrastructure-less virtual device composition and management do not consider such adaptation. We present a network performance influenced QoS model and its application for the graceful degradation and upgrade of virtual devices post-composition in MANETs, and through simulation show its effectiveness and efficiency.

8.1 Introduction

In Chapters 4-8 we addressed an open problem in achieving virtual devices, which is how to allow for the automatic composition of devices, and the services they provide, to create dynamic service configurations with limited human intervention. To this end, we

have derived a Service Overlay Network (SON) based definition of a virtual device applied to an autonomous policy driven framework for virtual device management, in addition to protocols and algorithms in achieving device/service selection, composition, and integration for the creation of a virtual device.

In an effort to cater to highly dynamic, mobile, and pervasive environments, our work has focused on virtual devices composed in ad hoc network environments. An ad hoc network is composed of a collection of wireless nodes whereby a portion or all may be mobile. Moreover, a wireless network is dynamically created among such nodes without aid from either infrastructure or administration. Hence, ad hoc networks can be described as self-creating, self-organizing, and self-administering; offering unique benefits and versatility to the virtual device problem. However, ad hoc networks create network environments whereby measurable attributes such as bandwidth, delay, loss, and jitter are highly likely to vary over time. With this in mind, a composed virtual device should gracefully degrade and upgrade along with the performance of the network. Moreover, this should be provided with minimal interruption to the user.

Lacking from our current solutions, as well as most MANET-based composition techniques, is consideration for such network-based attributes and respective adaptation. In this Chapter, we consider the modification of our existing virtual device composition solution to include network attributes into the service selection and composition process, as well as an extension to account for post-composition adaptation based on changing network attributes. Specifically, the contributions of this Chapter are the design and application of a network performance influenced QoS model for the graceful degradation and upgrade of virtual devices post-composition. Moreover, through simulation we examine the effectiveness and cost of our model.

The rest of this Chapter is organized as follows. Section 8.2 defines and applies a new QoS model for automated adaptation and Section 8.3 produces and discusses our simulation results. Section 8.4 gives our summary.

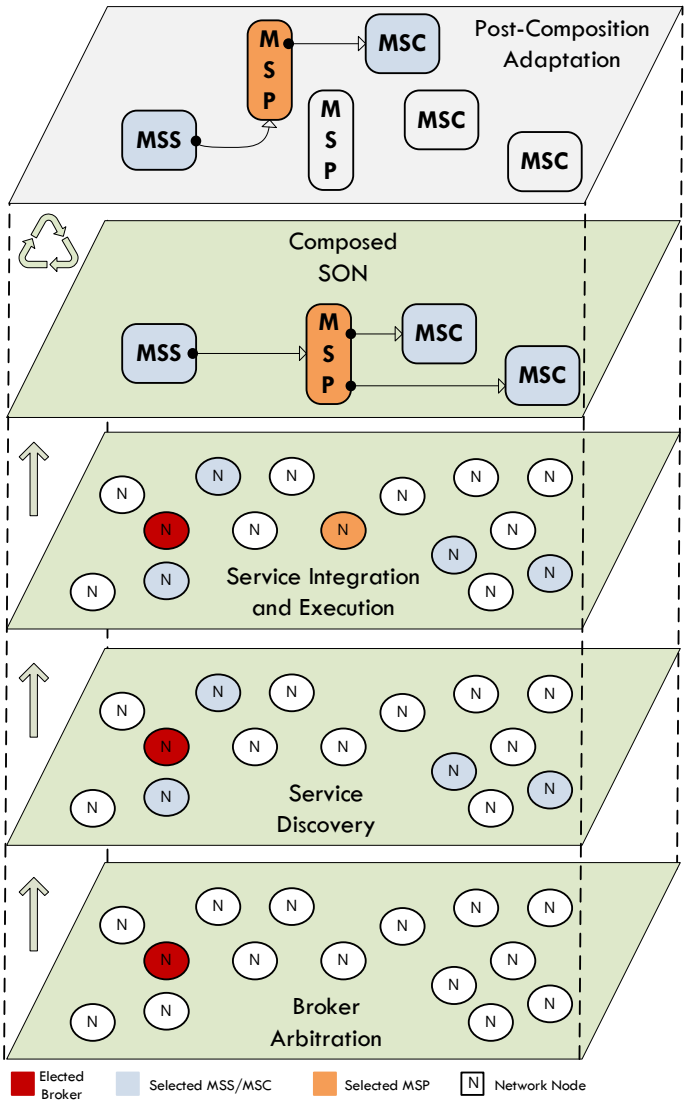


Fig. 8.1. The three phases of composition illustrated with extended automated adaptation.

8.2 Automated Adaptation

Virtual device composition in our framework is the product of a three phase process (illustrated in Figure 8.1) including broker arbitration, service discovery, and service integration and execution. *Broker arbitration* is the election of a node in the network with adequate resources (e.g. processor, power, load) to act as a broker in the composition process. The broker is responsible for initiating and managing the remaining two phases. *Service discovery* involves the discovery and selection of all

necessary end services, i.e. MSSs and MSCs, which satisfy the requirements of a particular TCspec. The selection of services for composition is based on the formation of service composition candidates, or sets of atomic discovered services satisfying the requirements of a particular TCspec. These composition candidates represent all the possible composition permutations available in the user's vicinity, whereby the best available composition candidate is selected for composition. S*ervice integration and execution* addresses the potential capability differences (e.g. supported codecs, resolutions, bandwidth requirements, etc.) occurring between MSSs and MSCs by discovering and integrating MSPs providing media processing functionality potentially requiring the splitting or joining of media. The result of this phase is a dynamic policy driven Service Overlay Network (SON) providing an end-to-end media delivery path to form a multimodal multi-device environment (i.e. virtual device).

As previously mentioned, ad hoc networks are known for their highly dynamic nature. Assuming that a composed virtual device will remain valid in a static form is incorrect. As the network changes, the virtual device should adapt to these changes in a manner providing minimal disruption to the user. In the following we utilize the measurable performance metrics of bandwidth, delay, loss, and jitter to develop a QoS model enabling the formed virtual device to flex with the performance variations of the network, through a process we refer to as automated adaptation adding a fourth phase to our framework (Figure 8.1).

Although, the process of adaptation begins after the initial creation of a virtual device, preparation for adaptation begins within the composition process itself. In our previous work, we developed two approaches to handle the broker arbitration and service discovery phases [Ch. 5 and 6] and one approach for handling the service integration and execution phase [Ch. 7]. At the heart of these approaches is a QoS model considering the imperative requirements of the virtual device concept, which consists of being aware of the user's continuously changing environment, immediately recognizing the expectations of the user, and instantiating a service with respect to the user's given circumstance. However, missing from this QoS model is the consideration for the current network performance and potential change in network performance.

8.2.1 Network Performance Influenced QoS Model

Our current QoS model identifies the degree of match between potential compositions and the requirements of a user's task (including service constraints satisfying both task requirements and user preferences) and the qualities and capabilities of service composition, taking into account the current state and availability of each service taking part in the composition. We now modify this model to also identify the necessary spectrum of tolerance in terms of bandwidth (BW), delay (D), loss (L), and jitter (J) that a particular composition can accept. Given such information, potential compositions, referred to as service composition candidates (SCCs), can be further classified by respective performance requirements. We formally represent this function as $QoS_{vd}[PR]$, representing the maximum QoS among all service composition candidates, satisfying the performance requirement *PR* where,

$$QoS_{vd}[PR] = Max(QoS_{scc})[PR], \quad \text{(A1)}$$

$$PR_{scc} = PR(\forall S_i \in SCC, Max(BW), Min(L), Min(D), Min(J)), \quad \text{(A2)}$$

$$QoS_{scc} = A_w \cdot \sum_j W_{tj,u} \cdot Sim_{tj}(d_j, d_{j,tcspec}), \quad \text{(A3)}$$

$$PR_{Si} = PR(S_i, BW_i, L_i, D_i, J_i), \quad \text{A4)}$$

$$A_w = Min(A_{ws}), \text{ and} \quad \text{A5)}$$

$$A_{ws} = \det avail(STs, Ps, Qs, ...). \quad \text{(A6)}$$

Using a similarity function *Sim* we are able to classify, in terms of qualities and capabilities, similarities between services in a TCspec and those available in the user's environment. Moreover, Sim_{Ti} is a precise similarity function for a specific service or service property d_j of type T_j. The output of the similarity function is a normalized value between [0,1], taking into deliberation both positive and negative discrepancies from a precise match, where the nearer the value is to 1, the higher the degree of match. In shaping the amount of relevance given to a particular service or service property type, each iteration of the summation is affected by a user-specific or default weight $W_{ti,u}$, representing the quantity of weight given to a particular service or service property type. Applied to the summation as a whole is a second weight A_w symbolizing the service availability of the potential service composition. A_w is attained by computing the service

availability A_{ws} for each service implicated in the potential composition, and computing the minimum (i.e. Equation (A5)). The calculation of A_{ws} is founded on various factors including the current state of the service ST_s, the usage policy of the service P_s, queue information Q_s regarding other users waiting for the service, and other information relevant to service availability (i.e. Equation (A6)). Every service S_i has a performance requirement formed as a bound PR_{Si} representing the minimum bandwidth and maximum loss, delay, and jitter it is able to tolerate (Equation (A4)). The performance requirement of a particular *SCC* can be computed by adopting the maximum *BW* and minimum *L*, *D*, and *J* values across all services involved in the *SCC* (Equation (A2)).

8.2.2 Application of the Model

Applying the preceding QoS model now assumes that an advertised service contains performance tolerance information, and that any particular node is able to obtain periodic performance metrics from the network. That being given, integrating the model into a composition scheme leads to the generation of a dynamic hash table HT_{vd} (shown in Figure 8.2) whereby,

- the *values* of the table represent discovered service composition candidates (*SCC*) currently available in the user's environment,
- the *keys* of the table represent the current performance requirement λ of the network, where $\lambda = (BW, L, D, J)$, and the *hash function H* maps a λ_i to a particular SCC_j, such that $H=QoS_{vd}[\lambda]$.

The completeness of HT_{vd} and the efficiency of obtaining HT_{vd} depends on whether a *push* or *pull* method is being used in terms of service advertisement. A push method involves nodes providing services continuously broadcasting service advertisement, and hence pushing their services to nearby nodes. In contrast, a pull method involves a node only advertising its services when a peer node shows interest, whereby the interested node pulls the advertisement to itself. A push-based composition technique leads to a largely complete HT_{vd}, such that a broker of a composition will continuously receive notices of new or modified services. However, this technique is inefficient as is leads to the unnecessary depletion of node resources (i.e. battery and processing power), such

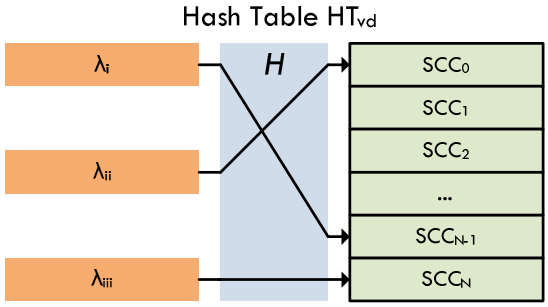


Fig. 8.2. Mapping network performance requirements to service composition candidates.

that service advertisements are received whether desired or not. A pull based composition is more efficient in obtaining HT_{vd}, such that the broker only fills HT_{vd} with services it is explicitly interested in at the moment. However, this directly results in an incomplete HT_{vd}, whereby after initial composition, unlike the push technique, the broker becomes unaware of the appearance of new or modifications of existing services in the network. This in turn requires additional work to be done, or even complete re-composition, in the presence of necessary adaptation. Therefore, a tradeoff exists.

In an effort to move away from broadcast-heavy composition techniques our current virtual device composition solution, VD-distCSP (Ch. 6), utilizes a pull technique, whereby we model and solve the virtual device problem as a *distributed constraint satisfaction problem (distCSP)* [116]. We now modify the problem such that *VD-DistCSP* $= (X, D, C)$ where:

- $X = \{Xi\}(i = 1,..., n; Xi \subseteq D)$. X_i is a variable corresponding to the set of services being provided by node *i* to satisfy a particular task.
- $D = \{Di\}(i = \{1,..., n\})$. *D* is a set of *n* domains of values, such that D_i is the service option domain of X_i.
- *C* is the set of constraints of the problem, which we now modify from our previous solution to include consideration for our newly defined QoS model. We can formulate *C* as follows:

$$\forall i, j \quad Xi \cap Xj = \{\ \}\ \ i, j \in \{1,...n\}, i \neq j \tag{B1}$$

$$\forall i \ \bigcup Xi = \{S\} \tag{B2}$$

$$PR(\{S\}, B, L, D, J) \geq \lambda \tag{B3}$$

$$\{S\} = Max(QoSscc) \quad \text{(B4)}$$

S is a set of services equivalent to all k required services d_j, $j = \{1,...,k\}$, where each service corresponds to a particular task requirement policy of a TCspec. Constraint (B1) guarantees that at most one node provides any particular service. Constraint (B2) ensures that all required services have been assigned. Constraint (B3) is added to ensure that the performance requirements of the composed service are more constrained than or equal to that of the current performance requirement λ of the network, and constraint (B4) enforces the fact that the chosen service set S is that maximizing QoS_{vd} [λ].

In our current solution, as a task becomes activated on a user's device, the respective node sends out a *virtual device request (VDrequest)* message into the user's immediate vicinity, using a bounded broadcast. A VDrequest contains the task's TCspec, which is used by a receiving node to decide whether or not it can contribute a service towards achieving task requirements. Given our new QoS model, a node able to contribute to a task also implies it's ability to meet the current performance requirement λ of the network. Such nodes send a *virtual device reply (VDreply)* message to the VDrequest initiator. The VDrequest initiator utilizes these replies to compute a *candidate table* containing the addresses and unique identifiers of all responding nodes and then proceeds to disseminate the table to all nodes identified in the table. Unlike our previous solution, the candidate table now only contains nodes able to meet the restrictions imposed by λ. Together these nodes execute the virtual device backtracking (VD-Btracking) algorithm, which is based on the traditional asynchronous bactracking algorithm for solving distCSPs [116]. The result of which, is a single service composition candidate maximizing QoS_{vd}[λ]. Additional details regarding the VD-Btracking algorithm can be found in.

Although our previous results showed the efficiency and low cost of forming a virtual device using our DistCSP-based solution using a pull method over a broadcast-based push method such as that of Chakraborty *et. al*. [65], our results did not consider post-composition adaptation. As previously mentioned, our DistCSP being a pull technique implies that should network conditions no longer satisfy the service composition initially selected, additional work will need to be done. In this case, as the broker periodically monitors network conditions, should a λ occur that the composed

virtual device cannot satisfy, or should λ allow for a service upgrade, a new virtual device request will need to be sent out into the user's vicinity forming a new candidate table and triggering another iteration of the VD-Btracking algorithm. This essentially involves the re-composition of the virtual device beginning at the service discovery phase.

In the following Section we analyze whether the additional work needed for adaptation, when using a pull-based technique still leads to a more efficient and cost-effective solution than utilizing a resource heavy broadcast-based push technique.

8.3 Simulation Results and Analysis

This section evaluates the performance of our VD-DistCSP protocol, using a pull method for composition, for different composite lengths in varying service densities, mobility, and topologies. We compare our results to those of D. Chakraborty *et. al.*'s Distributed Service Composition protocol (DSC) [65], using a push method for composition and discussed in Section 8.2, which we modify to include our network performance influenced QoS model $QoS_{vd}[PR]$.

8.3.1 Simulation Parameters and Metrics

We construct a MANET and implement the VD-DistCSP and DSC protocols using J-SIM [120], a well recognized simulation environment. In an attempt to simulate a home environment, we employ an area of 30x30m enclosing 25 nodes with a transmission range of 30m. A random-way-point mobility model is adopted with a minimum speed of 1m/s, a maximum speed of 3m/s, and a 5s stoppage time. All broadcasts are constrained to a strict bounded hop count of 1, and the DSC service advertising intervals are set to 5s. Our simulations are constructed to mirror a mobile device entering a home environment and forming a virtual device, followed by 100 simulated changes in the network performance metrics of bandwidth, delay, loss, and jitter forcing the formed virtual device to make 100 respective and consecutive adaptations. The simulation is repeated for composite lengths 3, 5, and 7, and service densities 20-100%. We define service

Duration	100 requests over 24 hour period
Space(x,y)	30×30 m
Transmission range	30 m
No. of nodes	25
Advertisement interval (DSC)	5 sec
Advertisement lifetime (DSC)	7 sec
Control hop count	1
Mobility	Random-way-point model with a minimum speed of 1m/s, a maximum speed of 3m/s, and a 5s stoppage time
Initial topology	Grid topology with nodes equally spaced out in (x,y)
Composite lengths	3, 5, and 7
Service density	20 to 100%

Table 8.1. Simulation parameter summary.

density as the percentage of nodes containing one of the atomic services required in the composition. Identified in each simulation is the achieved QoS as a percentage of the best QoS currently available based on network performance, and the amount of time and number of messages consumed in the process of adaptation.

8.3.2 Simulation Analysis

From Figure 8.3, results indicate that both VD-DistCSP and DSC are able to continuously discover the optimal composition adaptation (i.e. lines overlap at 100%) for varying service densities and composition lengths. This is expected as the two protocols are using identical QoS models. The differences in the protocols are visible in terms of the cost accrued in achieving adaptation.

Taken as an average per single adaptation, the number of messages used by VD-DistCSP (Figure 8.4) is considerably lower than that of DSC, using an average of 89% fewer messages. However, an average composition time decrease by DSC of 3.05 seconds is present when compared to VD-DistCSP (Figure 8.5), such that DSC holds at

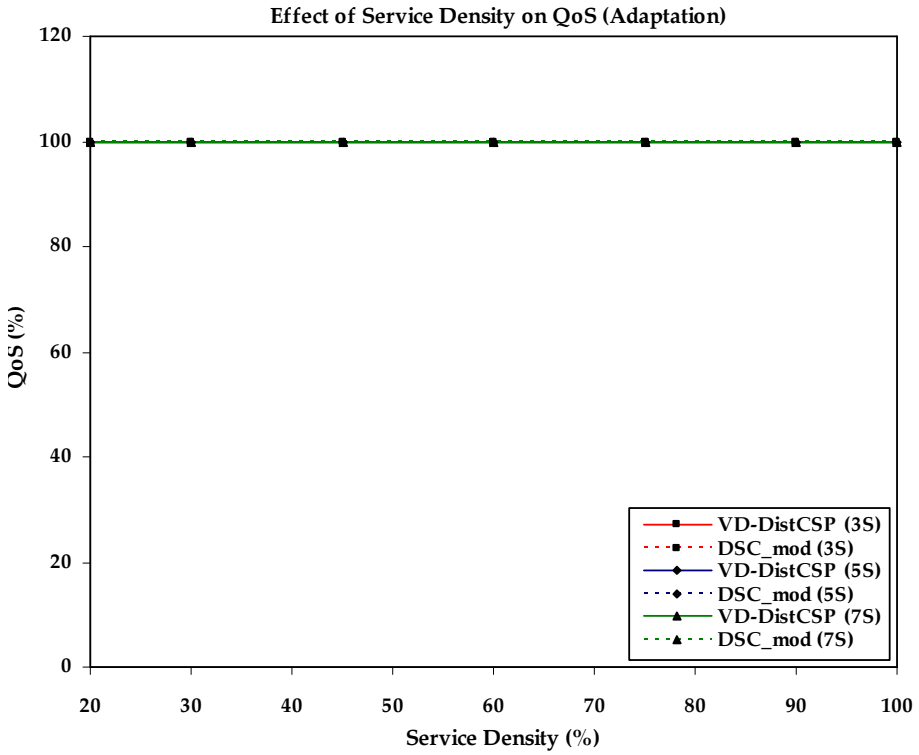


Fig. 8.3. QoS with respect to service density for composition lengths 3, 5, 7.

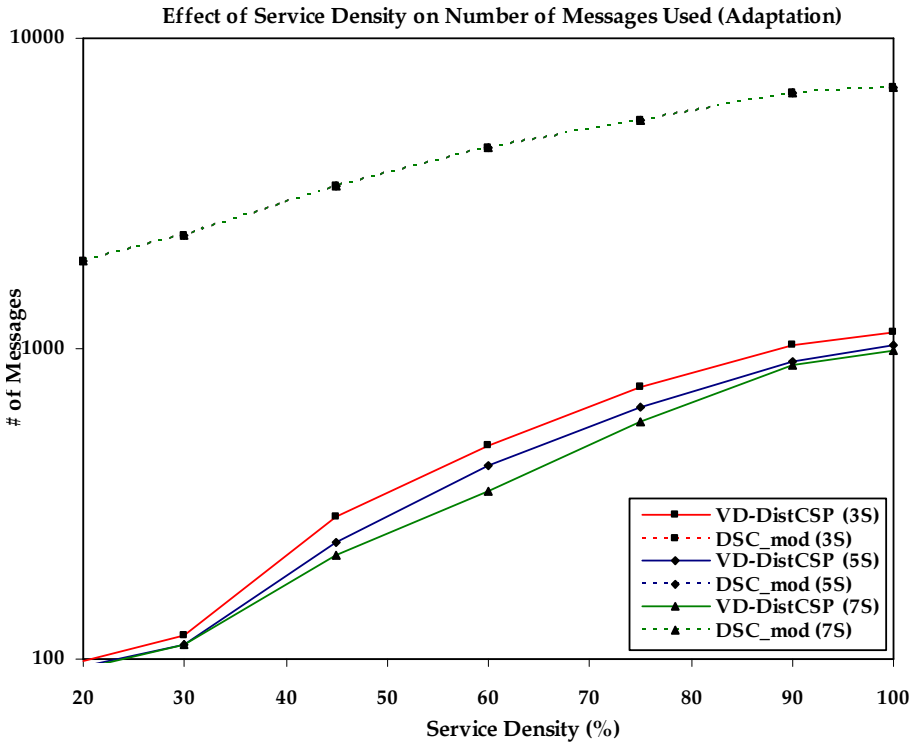


Fig. 8.4. Number of messages consumed with respect to service density for composition lengths 3, 5, 7.

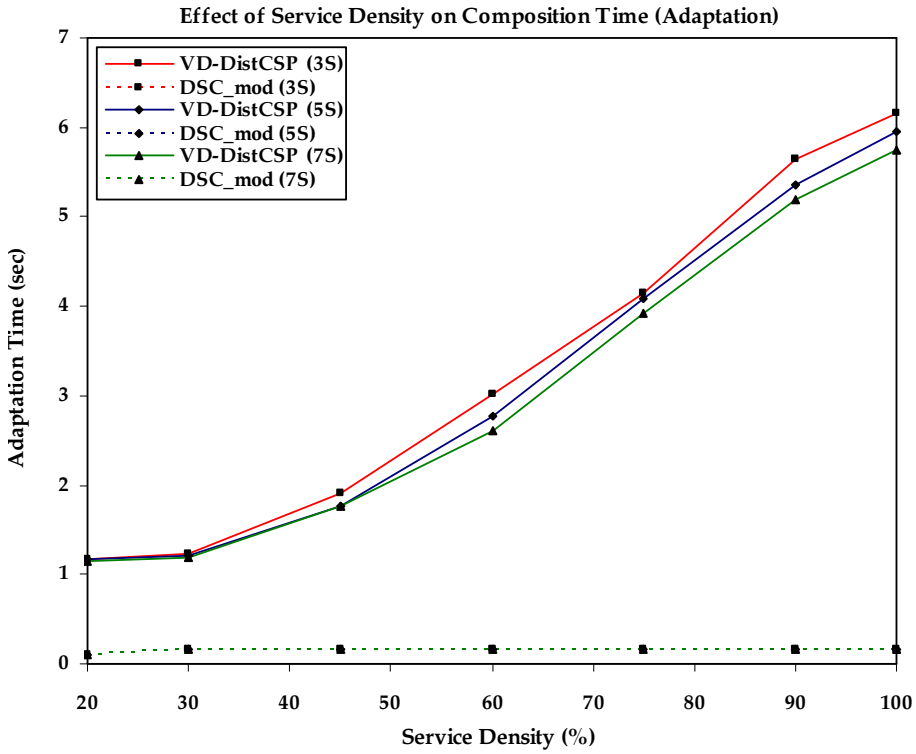


Fig. 8.5. Adaptation time with respect to service density for composition lengths 3, 5, 7.

about 0.15 seconds for all service densities and composition lengths shown in Figure 8.5. The results show a visual example of the tradeoff that exist between the push and pull methods. As DSC utilizes a push method, the broker responsible for composition is continuously and periodically receiving service advertisements from service providing nodes in the vicinity, the result of which can be seen in Figure 8.4 such that the higher the service density, i.e. the more advertising nodes, the higher the number of messages used. The benefits of this technique are clearly portrayed in Figure 8.5, such that, using DSC, the broker's HT_{vd} is highly complete, hardly requiring any time at all to initiate the optimal adaptation of the virtual device. In contrast, the time required for VD-DistCSP to adapt is directly relative to service density and composition length, whereby as the performance metrics of the network change, re-composition is necessary in order for the broker to acquire sufficient information to initate the optimal adaptation. This is also visible in Figure 8.4, whereby, using a pull method, the process of composition is significantly more efficient than DSC, not requiring continuous and periodic service advertisements.

From our results, as well as those presented in Ch. 6, we find that our VD-DistCSP protocol would benefit from small amounts of pushed service advertisements. That is to say that a hybrid push/pull technique would potentially offer better results. Such a technique could involve the single broadcast of service advertisements, as opposed to continuous periodic broadcasts, when a service is introduced, modified, or about to be deleted from the environment. This would allow the broker to keep a more complete HT_{vd} minimizing the frequency of necessary re-composition, without inducing heaving amounts of resource loss. Moreover, this would allow the protocol to better react to scalability and mobility.

As the mechanisms for discovering services in both VD-DistCSP and DSC remain unchanged, results in terms of scalability and mobility remain identical to those presented in Chapter 6.

8.4 Summary

In this Chapter, we have presented a network performance influenced QoS model for the graceful degradation and upgrade of a virtual device post-composition. In doing so, we pointed out the issues missing from current schemes as well as our own previous work, and moreover, presented the application of the new model into our distCSP-based approach for virtual device composition. Identified in discussion and through simulation, were set-backs of the technique with respect to push and pull based service advertisement schemes.

Chapter 9

Conclusions and Future Research Directions

This Chapter identifies contributed research work and discusses planned and future directions. The Chapter is organized as follows. Section 7.1 provides a summary of conducted research contributions in the area of dynamic composition and management of virtual devices for ad hoc multimedia service delivery. A summary of future research directions is then provided in Section 7.2.

9.1 Thesis Contributions

The focus of the dissertation research has been the development of a dynamic composition and management scheme for virtual devices. As the virtual device is not yet well defined in research literature, the first step towards this focus included a thorough literature study of media service delivery and the various aspects of the virtual device. More precisely, we aimed to address two questions: what are the requirements of virtual device composition and management, and why is it difficult to satisfy these requirements with current approaches? The result of this phase has been a state-of-the-art survey of major research directions and efforts in the areas of media delivery, service oriented pervasive computing, service discovery, and service composition. Based on the

identified limitations of current research work, a novel framework for the autonomous management of virtual devices has been designed. In the following, a summary of the main contributions of our research work is given.

- A novel SON-based definition of a virtual device applied to an autonomous policy driven framework for virtual device management formed as a hierarchical structure of distributed elements, including autonomic elements, all working towards the self-management of virtual devices. This includes the dynamic discovery, selection, and composition of multimodal multi-device services present in a user's changing environment, whereby the fusing of multimodal multi-device services lies in the self-configuration of service overlay networks.
- A distributed broker-based service composition protocol tailored for virtual device composition, where the elected broker is not only resource rich and located in a high service density region, but located in a region providing the greatest opportunity for discovering services providing the highest degree of match with regards to a user's desired task. Moreover, infused into the protocol is a means of identifying the variation of similarity between two services of the same type, the current availability of each service, and how important such variation and availability is to the user, for a service composition that forms the best possible virtual device at the time of need. Simulation results demonstrated the effectiveness of the extension and low cost, in terms of increased amounts of messages and composition time, imposed for added functionality.
- A distributed constraint satisfaction problem (distCSP) model for task-based, quality-of-service aware, virtual device composition in MANETs, and a virtual device distCSP protocol utilizing an asynchronous backtracking-based algorithm for solving the respective distCSP. Simulation results show the effectiveness of our method in performing virtual device composition without the use of broadcast-based service advertisements, whereby works relying on discovery mechanisms largely based on constant periodic service advertising by controlled broadcast unnecessarily deplete node resources (i.e. battery and processing power).
- An extension to the distCSP model for capability reconciliation in MANETs and a respective protocol utilizing an asynchronous backtracking-based algorithm for

solving the particular distCSP, for capability reconciliation in MANETs. Capability reconciliation is a requirement such that the dynamic and ad hoc nature of the discovery and composition of such devices and the services they provide inevitably leads to capability differences, whereby the input of a service B is not compatible with the output of a service A; A and B needing to be composed. Simulation results show the effectiveness and efficiency of the solution.

- A design and application of a network performance influenced QoS model for the graceful degradation and upgrade of virtual devices post-composition, such that assuming that a composed virtual device is to remain static in nature once generated is false, as the virtual device should gracefully degrade and upgrade along with the conditions available in the user's environment; particularly, the current performance requirements of the network. Simulation results show the effectiveness and cost of our model.

9.2 Future Research Directions

The following provides the main focus of our future research work, which is based on:

- The continued investigation of autonomous policy-based management as a means of constructing and maintaining virtual devices;
- Increased experimentation using multiple Media Service Clients in the capability reconciliation process;
- The development of a hybrid push/pull service advertisement mechanism;
- The investigation of a moving virtual device; and,
- Further fulfillment of autonomic properties.

The following Section discusses in further detail these proposed research directions.

9.2.1 Automated Policy Management for SON-based Virtual Devices

Introduced in Chapter 3 was an autonomous policy driven framework for virtual device management. Although a significant amount of research has been carried out in the area

of policy-based management [123-125], existing techniques mainly focus on the defining of priori policy configurations to manage network devices. Such approaches lead policies to be static in nature and do not cater to the dynamic nature of the virtual device nor the "zero touch" user aspect. Therefore, we investigate the design of a virtual device specific hierarchical policy model used in facilitating the mapping of higher level abstract user/application policies into lower level objectives used in virtual device configuration, and management. Given sets of constraints, objectives, and sets of possible actions to be taken, decisions for policy customizations are taken at runtime. By integrating autonomic overlays as a means of media delivery management as well as dynamic policy creation, we begin to examine QoS provisioning from more of a network level perspective, whereby QoS now goes beyond discovering and composing appropriate services in the user's environment, but includes network level objectives such as delay, throughput, and error rates. In this manner overlay networks self configure and adapt to ensure the satisfaction of adequate QoS for the lifetime of the virtual device. Achieving such an end requires the investigation of autonomous overlay management driven by dynamic policy creation.

9.2.2 Multiple Media Service Clients

Experiments conducted in Chapter 7 were limited to capability reconciliation with a single MSC. Handling multiple MSCs can be dealt with in a similar fashion, whereby each *comPath* now holds a unique identifier, such that when a path is split into two or more separate sub-paths each sub-path retains this identifier. Moreover, included with the sub-path is the pre-split $\beta_{current}$ value, whereby β_{total} is the sum of the $\beta_{sub\text{-}paths}$ added to $\beta_{pre\text{-}split}$. Each node completing the composition of a sub-path is responsible for finding the minimum $\beta_{sub\text{-}path,}$ and the highest priority node completing the composition of a sub-path is responsible for representing the overall *comPath* in the negotiation process. Further experimentation should be conducted to identify the strengths and weaknesses of such an approach. Moreover, such work should be combined with the further investigation of policy management.

9.2.3 Hybrid Push/Pull Service Advertisement

From results displayed in Chapter 6 and 8 we found that our VD-DistCSP protocol would benefit from small amounts of pushed service advertisements. As is, the VD-DistCSP protocol is best suited for personal network type environments, such as the home or office. Chapter 6 pointed out scalability issues that are not part of such an environmental scope, but that should be addressed none the less. Moreover, results in Chapter 8 showed that post-composition adaptation is constrained by the lack of continuous real time knowledge provided by push-based service advertisements. As a result, it can be concluded that a push/pull technique would potentially offer better results. Such a technique could involve the single broadcast of service advertisements, as opposed to continuous periodic broadcasts, when a service is introduced, modified, or about to be deleted from the environment. This would allow the broker to stay knowledgeable without inducing heaving amounts of resource loss.

9.2.4 A Moving Virtual Device

Although the solutions provided in this thesis account for the movement of nodes within a user's proximity, not considered was the movement of the virtual device itself. Further research is required accounting for a scenario whereby a user composes a virtual device within one environment, and then proceeds to move out of the environment into another. It is a reasonable assumption that some components of the composed virtual device would move with the user, but not all. Therefore, further services satisfying the virtual device may need to be discovered and a hand-off procedure identified, which includes the potential hand-off between two or more brokers.

9.2.5 Autonomic Properties

Chapter 1 provides an outline of our architecture containing autonomic elements contributing the attributes necessary for self-management. However, the solution presented in this thesis does not provide all the attributes of autonomic computing,

which are self-configuration, self-optimization, self-healing, and self-protection [98] [114]. The self-protection property implies the system's ability to adapt to a dynamically changing environment; self-optimization allows a system to tune resources and balance workload to maximize their use; self-healing is the ability to discover, diagnose, and act to prevent disruptions; and, self-protection enables the anticipation, detection, identification, and protection against attacks. Aspects of self-configuration and self-optimization were provided in this thesis; however, aspects of self-healing and self-protection, such as dealing with non-compliant or malicious nodes and services were not. Further research should be undertaken to account for these remaining properties.

List of Publications

Journal Publications

1. E. Karmouch and A. Nayak, "Capability Reconciliation for a CSP Approach to Virtual Device Composition," submitted to the *IEEE/ACM Transactions on Networking*, January 2011.
2. E. Karmouch and A. Nayak, "Distributed Virtual Device Composition in Mobile Ad Hoc Networks," submitted to the *IEEE Transactions on Mobile Computing*, December 2010.
3. E. Karmouch and A. Nayak, "A Distributed Constraint Satisfaction Problem Approach to Virtual Device Composition," submitted to the *IEEE Transactions on Parallel and Distributed Systems*, December 2010.
4. N. Goel, K. Kalaichelvan, E. Karmouch, A. Nayak, I. Stojmenovic, and E. Villanueva-Pena, "Physical Layer Impact on Finding Local Knowledge Information in Ad Hoc and Sensor Networks," *International Journal of Computational Science*. Vol. 2, No. 2, pages: 233-249, 2008.

Conference Publications

1. E. Karmouch and A. Nayak “Capability Reconciliation for Virtual Device composition in Mobile Ad Hoc Networks," in the *Proceedings of the 6th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2010)*, October 2010, pages: 27-34. [Best Paper Award]
2. E. Karmouch and A. Nayak "Towards the Autonomous Management of Virtual Devices," in the *Proceedings of the IEEE Workshop on Enabling the Future Service-Oriented Internet: Towards Socially-Aware Networks (EFSOI'09)*, Dec ember 2009, pages: 1-5.
3. E. Karmouch and A. Nayak "A Distributed Constraint Satisfaction Problem for Virtual Device Composition in Mobile Ad Hoc Networks," in the *Proceedings of the IEEE Global Communications Conference (GLOBECOM'09)*, December 2009, pages: 1-7.
4. E. Karmouch and A. Nayak "A Distributed Protocol for Virtual Device Composition in Mobile Ad Hoc Networks," in the *Proceedings of the IEEE International Conference on Communications (ICC'09)*, June 2009, pages: 1-6.

Bibliography

[1] M. Alcaniz and B. Rey, “New Technologies for Ambient Intelligence,” Ambient Intelligence, IOS Press, 2005, pp 3-15.

[2] M. Merabti, P. Fergus, O. Abuelma'atti, H. Yu, and C. Judice, “Managing Distributed Networked Appliances in Home Networks,” in *Proceedings of the IEEE, Special Issue on Recent Advances in Distributed Multimedia Communications*, vol. 96(1), 2008, pp. 166-185.

[3] Networked Media Task Force, “Networked Media of the future,” October 2007. [Online]. Available: ftp://ftp.cordis.europa.eu/pub/fp7/ict/docs/netmedia/networked-media-of-the-future_en.pdf. [Accessed March 2010].

[4] Media Delivery Platforms Cluster, “Multimedia Delivery in the Future Internet: A Converged Network Perspective,” White Paper, October 2008.

[5] W. Buxton, “Less is More (More or Less),” in *The Invisible Future: The seamless integration of technology in everyday life*, P. Denning (Ed.), New York: McGraw Hill, 2001, pp. 145-179.

[6] M.P. Papazoglou and G. Georgakapoulos, “Service-oriented computing ,” *Communications of the ACM*, Vol. 46, No. 10, October 2003.

[7] J. P. Sousa and D. Garlan, “Aura: an architectural framework for user mobility in ubiquitous computing environments,” In the *Proceedings of the IFIP 17th World Computer Congress - TC2 Stream / 3rd IEEE/IFIP Conference on Software Architecture*, 2002, pp. 1-14.

[8] M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, R.H. Campbell, and K. Nahrstedt, “Gaia: a middleware platform for active spaces,” *SIG-MOBILE Mobile Computing and Communication Review*, Vol. 6, No.4, 2002.

[9] Kumar, M., Shirazi, B. A., Das, S. K., Sung, B. Y., Levine, D., and Singhal, M., “Pico: A middleware framework for pervasive computing.,” IEEE Pervasive Computing, 2003.

[10] Issarny, V., Sacchetti, D., Tartanoglu, F., Sailhan, F., Chibout, R., Levy, N., and Talamona, A., "Developing ambient intelligence systems: A solution based on web services," Journal of Automated Software Engineering, 2005.

[11] Computer Science and Artificial Intelligence Laboratory, "MIT Project Oxygen," [online] Project Overview, available from http://oxygen.lcs.mit.edu/ [Accessed January 2010].

[12] K. Arnold, B. Osullivan, R. W. Scheifler, J. Waldo, and A. Wollrath., "The Jini Specification (The Jini Technology).,"Addison-Wesley, Reading, MA, June 1999.

[13] The Salutation Consortium Inc, "Salutation Architecture Specification (Part 1) Version 2.1," [online] available from http://systems.cs.colorado.edu/grunwald/MobileComputing/Papers/Salutation/Sa20e1a21.pdf [Accessed January 2010].

[14] R. John, "UPnP, Jini and Salutaion - A Look at some popular Coordination Frameworks for Future Network Devices," Technical report, California Software Labs, 1999.

[15] E. Guttman, C. Perkins, and J. Veizades, "Service Location Protocol," RFC 2165, June 1997.

[16] S. E. Czerwinski, B.Y. Zhao, T. D. Hodes, A. D. Joseph, and R. H. Katz, "An Architecture for a Secure Service Discovery Service," In the *Proceedings of the Fifth International Conference of Mobile Computing and Networks*, 1999.

[17] F. Casati, S. Ilnicki, L. Jin, V. Krishnamoorthy, and M. Shan, "Adaptive and Dynamic Service Composition in eFlow," In the Technical Report, HPL-200039, Software Technology Laboratory, Palo Alto, California, USA, March 2000.

[18] N. Niebert, A. Schieder, H. Abramowicz, G. Malmgren, J. Sachs, U. Horn, C. Prehofer, and H. Karl, "Ambient networks: an architecture for communication networks beyond 3G," *IEEE Wireless Communications*, Vol.11, No.2, Apr 2004, pp. 14-22.

[19] F.Hartung, S.Herborn, M.Kampmann, and S.Schmid, "Smart Multimedia Routing and Adaptation using Service Specific Overlay Networks in the Ambient Networks Framework," WWRF #12 , Nov 4-5, 2004.

[20] ITU, “Broadband Mobile Communications Towards a Converged World,” *ITU/MIC workshop on shaping the future mobile information society*, document: SMIS/05, February 19, 2004.

[21] L. Popescu, “Supporting Multimedia Session Mobility using SIP,” in the *Proceedings of Communication Networks and Services Research Conference*, 2003, pp. 112-123.

[22] Q. Wang, and M. A. Abu-Rgheff, “Next-Generation Mobility Support,” *Communications Engineer*, Vol. 1, No. 1, February 2003, pp 16-19.

[23] C. Perkins, “Mobile Networking Through Mobile IP,” *IEEE Internet Computing*, January 1998, pp. 58-69.

[24] C. Perkins, “IP Mobility Support for IPv4,” *IETF RFC 3344*, August 2002.

[25] A. Misra, S. Das, A. Dutta, A. McAuley, S. K. Das, "IDMP-based fast handoffs and paging in IP-based 4G mobile networks ," *IEEE Communications Magazine*, Vol.40, No.3, Mar 2002, pp.138-145.

[26] A. T. Campbell, J. Gomez, S. Kim, and A.G. Valkó, “Design, Implementation, and Evaluation of Cellular IP,” *IEEE Personal Communications*, August 2000, pp. 42-49.

[27] R. Ramjee, K. Varadhan, L. Salgarelli, S. R. Thuel, S. Y. Wang, and T. La Porta, "HAWAII: a domain-based approach for supporting mobility in wide-area wireless networks," *IEEE/ACM Transactions on Networking*, Vol. 10, No. 3, June 2002, pp.396-410.

[28] C. Perkins, “IP Encapsulation Within IP,” *IETF RFC 2003*, October 1996.

[29] H. Schlzrinne, and E. Wedlund, “Application-Layer Mobility Using SIP,” *ACM MC2R*, Vol. 4, No. 3, July 2000, pp. 47-57.

[30] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, “SIP: Session Initiation Protocol,” RFC 3261, June 2002.

[31] J. Rosenberg, D. Willis, R. Sparks, B. Campbell, H. Schulzrinne, J. Lennox, C. Huitema, B. Aboba, D. Gurle, and D. Oran, “SIP extensions for presence,” IETF Internet Draft, June 2000.

[32] A. Niemi, “Session Initiation Protocol (SIP) Extension for Event State Publication,” IETF RFC3903, October 2004.

[33] J. Rosenberg, J. Peterson, H. Schulzrinne, and G. Camarillo, "Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP)," IETF RFC3725, April 2004.

[34] R. Sparks, "The Session Initiation Protocol (SIP) Refer Method," IETF RFC3515, April 2003.

[35] C. Politis, K. A. Chew, and R. Tafazolli, “Multilayer Mobility Management for All-IP Networks: Pure SIP vs. Hybrid SIP/Mobile IP,” in the *Proceedings of 57th IEEE Semiannual Vehicular Technology Conference*, April 2003.

[36] TS 23.228, “IP Multimedia Subsystem (IMS),” 3GPP, Release 6.

[37] P. Kim and W. Boehm, “Support for Real-Time Applications in Future Mobile Networks: the IMS Approach,” in the *Proceedings of International Symposium on Wireless Personal Multimedia Communications*, Oct. 2003.

[38] T. Renier, L. KimLynggarg, G. Castro, and H. P. Schwefel, "Mid-Session Macro-Mobility in IMS-Based Networks," *IEEE Vehicular Technology Magazine*, Vol. 2, No. 1, March 2007, pp. 20-27.

[39] K. L.Johnson, J. F. Carr, M. S. Day, and M. F. Kaashoek, “The Measured Performance of Content Distribution Networks,” in the *Proceedings of the Fifth WCW*, May 2000.

[40] D. Kaye, *Strategies for Web Hosting and Managed Services*, John Wiley & Sons, 2002.

[41] ITU-T, “Terms and definitions related to quality of service and network performance including dependability,” E.800, CCITT/ITU, August 1994.

[42] V.T. Raisinghani and S. Iyer, “Cross Layer Design Optimizations in Wireless Protocols Stacks,” *Computer Communications*, Vol. 27, No. 8, May 2004, pp. 720-724.

[43] I. Kofler, C. Timmerer, H. Hellwagner, and T. Ahmed, “Towards MPEG-21-based Cross-layer Multimedia Content Adaptation,” in the *Proceedings of the 2nd International Workshop on Semantic Media Adaptation and Personalization*, December 2007, pp 3-8.

[44] W. T. Ooi, R. V. Renesse, and B. Smith, “The design and implementation of programmable media gateways,” in the Proceedings of International Workshop on Network and Operating Systems Support for Digital Audio and Video, June 2000.

[45] M. Kampmann, M. Vorwerk, M. Kleis, S. Schmid, S. Herborn, R. Agüero, and J. Choque, “A Multimedia Delivery Framework for Ambient Networks,” WRF#15, December 2005.

[46] A. Schmidt, “Interactive Context-Aware Systems Interacting with Ambient Intelligence,” Ambient Intelligence, IOS Press, 2005, pp. 159 – 178.

[47] Marc Weiser, “Some Computer Science Issues in Ubiquitous Computing,” *Communications of the ACM*, Vol. 36, No.7 July 1993, pp.74-83.

[48] S. B. Mohktar, “Intergiciel S´emantique pour les Services de l’Informatique Diffuse,” Ph.D. Thesis, Université Pierre et Marie Curie – Paris 6, December 2007.

[49] F. Casati, M. C. Shan, D. Georgakopoulos, “E-Services - Guest editorial,” *VLDB Journal*, Vol. 10, No. 1, 2001.

[50] G. Weikum, “Special Issue on Infrastructure for Advanced e-Services,” *IEEE Data Engineering Bulletin*, Vol. 24, No.1, March 2001.

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

[52] R.H. Katz, E. A. Brewer, and Z.M. Mao, “Fault-tolerant, Scalable, Wide-Area Internet Service Composition,” Technical Report, UCB/CSD-1-1129, CS Division, EECS Department, UC, Berkeley, January 2001.

[53] J. N. Kok and K. Sere, “Distributed Service Composition,” In Technical Report No. 256, Turku Centre for Computer Science, Finland, March 1999.

[54] P. Queloz and A. Villazon, “Composition of Services with Mobile Code,” In the *Proceedings of First International Symposium on Agent Systems and Applications Third International Symposium on Mobile Agents*, 1999.

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

[56] WSDL, “Web Services Description Language 1.1.,” [online] available from http://www.w3.org/TR/wsdl [Accessed January 2010].

[57] DARPA, “Agent Markup Language for Services Specification Draft 0.5.,” [online] available from http://www.daml.org/services/daml-s/2001/05/ [Accessed January 2010].

[58] W3C, “OWL Web Ontology Language Overview” [online] available from http://www.w3.org/TR/owl-features/ [Accessed January 2010].

[59] WSFL, “Web Services Flow Language,” [online] available from http://xml.coverpages.org/wsfl.html [Accessed January 2010].

[60] IBM, “Business Process Execution Language for Web Services version 1.1,” [online] available from http://www.ibm.com/developerworks/library/specification/ws-bpel/ [Accessed January 2010].

[61] K. Erol, J. Hendler, and D. Nau, “HTN planning: Complexity and expressivity,” In the *Proceedings of the International Conference on Artificial Intelligence*, 1994.

[62] M. Paolucci, A. Ankolekar, N. Srinivasan, and K. Sycara, “The daml-s virtual machine,” In the Proceedings of the 2nd International Semantic Web Conference (ISWC), October 2003.

[63] S. Helal, N. Desai, and C. L. Konark, “A Service Discovery and Delivery Protocol for Ad-hoc Networks,” In the *Proceedings of the Third IEEE Conference on Wireless Communication Networks*, March 2003.

[64] D. Tang, C. Chang, K. Tanaka, and M. Baker, “Resource Discovery in Ad hoc Networks,” Technical report, CSL-TR-98-769, Stanford University, August 1998.

[65] D. Chakraborty, Y. Yesha, and A. Joshi, “A Distributed Service Composition Protocol for Pervasive Environments” in the *Proceedings of the IEEE Wireless Communications and Networking Conference*, Vol. 4, 2004, pp. 2575 - 2580.

[66] Bluetooth Specification. [online] available from http://www.bluetooth.org/. [Accessed March 2010]

[67] M. Klemettinen, *Enabling Technologies for Mobile Services: The MobiLife Book.* Wiley, 2007.

[68] R. Want, T. Pering, S. Sud, and B. Rosario, “Dynamic Composable Computing,” in the *Proceedings of the 9th Workshop on Mobile Computing Systems and Applications*, 2008, pp. 17-21.

[69] C. Namman, A. Mingkhwan, O. Abuelma’atti, and M. Merabti, “The Flexible Service Composition Framework for Networked Appliances,” in the *Proceedings of the International Conference on Innovations in Information Technology*, 2007, pp. 233-237.

[70] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris , “Resilient Overlay Networks,” In the *Proceedings of the 18th ACM Symp. on Operating Systems Principles*, Oct. 2001, pp. 131-145.

[71] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and J. O'Toole, “Overcast: reliable multicasting with an overlay network,” in the Proceedings of USENIX OSDI, Oct. 2000, pp.14-14.

[72] L. Subramanian, I. Stoica, H. Balakrishnan, and R. Katz, “OverQoS: An overlay based architecture for enhancing Internet QoS,” in the Proceedings of the Symposium on Networked Systems Design and Implementation, March 2004.

[73] Y. H. Chu, S. G. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," *Selected Areas in Communications, IEEE Journal on* , Vol.20, No.8, Oct 2002, pp. 1456-1471.

[74] B. Zhang, S Jamin, and L. Zhang, "Host multicast: a framework for delivering multicast to end users," in the *Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, Vol. 3, June 2002, pp. 1366-1375.

[75] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, “Chord: A scalable peer-to-peer lookup service for internet applications,” in the *Proceedings of ACM SIGCOMM*, August 2001, pp. 149 - 160.

[76] A.Nakao, L.Peterson, and A.Bavier, "A routing underlay for overlay networks," in the *Proceedings of ACM SIGCOMM*, Aug. 2003, pp. 11-18.

[77] K. Shen, “Saxons: Structure management for scalable overlay service construction,” In the *Proceedings of USENIX NSDI*, 2004, pp. 281–294.

[78] P. Francis, “Yoid: Extending the internet multicast architecture,” technical report, ACIRI, April 2, 2000.

[79] Planetary Network Testbed, [online] available from http://www.planet-lab.org. [Accessed March 2010].

[80] R. Braynard, D. Kostic, A. Rodriguez, J. Chase, and A. Vahdat, "Opus: an overlay peer utility service," in the *Proceedings of the IEEE Open Architectures and Network Programming*, 2002, pp. 167-178.

[81] Xbone, [online] available from http://www.isi.edu/xbone. [Accessed March 2010]

[82] L. Subramanian, I. Stoica, H. Balakrishnan, and R.H.Katz, “Overqos: Offering Internet QoS Using Overlays,” *SIGCOMM Comput. Commun. Rev.*, Vol. 33, No. 1, January 2003, pp. 11-16.

[83] K.Yang, A.Galis, T.Mota, and S.Gouveris, “Automated Management of IP Networks through Policy and Mobile agents,” in the *Proceedings of the 4th Int. Workshop on Mobile Agents for Telecommunication Applications*, Oct. 2002. pp. 249-258.

[84] N.Damianou, N.Dulay, E.Lupu, and M.Sloman, “The Ponder Specification Language,” Lecture Notes in Computer Science, Springer, Vol. 1995/2001. pp. 18-38.

[85] K. L. Calvert, “An Architectural Framework for Active Networks,” DARPA active nets document, 2001. [online] available from http://protocols.netlab.uky.edu/~calvert [Accessed January 2010].

[86] M. Solarski, and E. Moeller, “Challenges in active service deployment,” in the *Proceedings of the First International Workshop on Active Network Technologies and Applications*, March 2002.

[87] A. Galis, S. Denazis, C. Brou, C. Klein (Eds.), *Programmable Networks for IP Service Deployment*, Artech House Books, 2004, ISBN 1-58053-745-6, p. 450.

[88] M.Solarski, M.Bossardt, and T.Becker, "Deployment and management of component-based services in active networks," *The International Journal of Computer and Telecommunications Networking*, Vol 50, No. 14, October 2006, pp. 1389-1286.

[89] C.Dhillon, M.Bond, J.Griffioen, and K.L.Calvert, "Building layered active services," *The International Journal of Computer and Telecommunications Networking*, Vol. 50 , No. 14, October 2006, pp. 2475 – 2487.

[90] G. Cortese, R. Fiutem, P. Cremonese, S. D'antonio, M. Esposito, S. P. Romano, and A. Diaconescu, "Cadenus: creation and deployment of end-user services in premium IP networks," *IEEE Communications Magazine*,Vol. 41, January 2003, pp. 54 – 60.

[91] S. Khaldoon, M. Damien, and L.Pascal, "A Scalable Middleware for Creating and Managing Autonomous Overlays," in the *Proceedings of the 2nd International Conference on Communication Systems Software and Middleware*, Jan. 2007, pp. 1-8.

[92] K.Ragab, N.Y.Horikoshi, H.Kuriyama, and K.Mori, "Autonomous Decentralized Community Communication for Information Dissemination", *IEEE CS Internet Computing magazine*, Vol.8, No. 3, May-June 2004, pp.29-36.

[93] K.Ragab, Y.Horikoshi, H.Kuriyama, and K.Mori, "Multi-layer Autonomous Community Overlay Network for Enhancing Communication Delay," in the *Proceedings of the Ninth International Symposium on Computers and Communications*, Vol.2, 2004, pp. 987-992.

[94] S.Ratnasamy, M.Handley, R.Karp, and S.Shenker, "Topologically-aware overlay construction and server selection," in the *Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*. Vol. 3, 2002, pp. 1190-1199.

[95] C. J. Lin, Y. T. Chang, S. C. Tsai, and C. F. Chou, "Distributed Social-based Overlay Adaptation for Unstructured P2P Networks," in the *Proceedings of the IEEE Global Internet Symposium*, 2007, pp. 1-6.

[96] J. A. Pouwelse, P. Garbacki, J. W. A. Bakker, J. Yang, A. Iosup, D. Epema, M.Reinders, M. R. van Steen, and H. J. Sips, "Tribler: A social-based based peer

to peer system," in the *Proceedings of the 5th Int'l Workshop on Peer-to-Peer Systems*, February 2006.

[97] P. Androutsos, D. Androutsos, and A. Venetsanopoulos, "Small World Distributed Access of Multimedia Data: an Indexing System that Mimics Social Acquaintance Networks," *IEEE Signal Processing Magazine*, Vol. 23, No.2, March 2006, pp. 142– 153.

[98] IBM Corporation, "An architectural blueprint for autonomic computing," White Paper, June 2006.

[99] R. Farha and A. Leon-Garcia, "Blueprint for an Autonomic Service Architecture" in the *Proceedings of the International Conference on Autonomic and Autonomous Systems*, 2006.

[100] E.Kasten and P.McKinley, "MESO: Supporting Online Decision Making in Autonomic Computing Systems," *IEEE Trans. on Knowledge And Data Engineering*, Vol. 19, No. 4, April 2007.

[101] R. M. Bahati, M.A. Bauer, E. M. Vieira, O. K. Baek, and A. Chang-Won, "Using policies to drive autonomic management," in the Proceedings of the International Symposium on the World of Wireless, Mobile and Multimedia Networks, June 2006, pp. 475 - 479.

[102] J. Pena, M. G. Hinchey, R. Sterritt, A. Ruiz-Cortes, and M. Resinas, "A Model-Driven Architecture Approach for Modeling, Specifying and Deploying Policies in Autonomous and Autonomic Systems," in the *Proceedings of the 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing*, September 2006, pp.19-30.

[103] P. McKinley, F. Samimi, J. Shapiro, and C. Tang, "Service Clouds: A Distributed Infrastructure for Constructing Autonomic Communication Services," in the *Proceedings of the 2nd IEEE Int. Symposium on Dependable, Autonomic and Secure Computing*, 2006, pp. 341-348.

[104] D. Xiangdong, S.Hariri, L.Xue, H.Chen, M.Zhang, S.Pavuluri, and S.Rao, "Autonomia: an autonomic computing environment," in the *Proceedings of the IEEE Int. Performance, Computing, and Communications Conf.*, April 2003, pp. 61–68.

[105] P. Grace, G. Coulson, G. Blair, L. Mathy, W. Yeung, W. Cai, D. Duce, and C. Cooper, “GRIDKIT: Pluggable Overlay Networks for Grid Computing,” in the *Proceedings of the Distributed Objects and Applications Conf.*, October 2004, pp. 1463-81.

[106] M. Parashar, H. Liu, Z. Li, V. Matossian, C. Schmidt, G. Zhang, and S. Hariri, “AutoMate: Enabling Autonomic Applications on the Grid,” *Cluster Computing*, Vol. 9, No. 6, 2006, pp. 161–174.

[107] D. M. Chess, A. Segal, I. Whalley, and S. R. White, "Unity: Experiences with a Prototype Autonomic Computing System," in the *Proceedings of the International Conference on Autonomic Computing*, 2004, pp. 140-147.

[108] S. Dobson, S. Denazis, A. Fernández, D. Gaïti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, F. Zambonelli, and A. Fernández, “A Survey of Autonomic Communications,” *ACM Transactions on. Autonomous and Adaptive Systems*, Vol. 1, No. 2, December 2006, pp. 223-259.

[109] A. Fox, B. Johanson, P. Hanrahan, and T. Winograd, “Integrating Information Appliances into an Interactive Workspace,” *IEEE Computer Graphics and Applications*, Vol. 20. No. 3, May 2000, pp54-65.

[110] B. A. Myers, "Using Hand-Held Devices and PCs Together," *Communications of the ACM*. Vol. 44, No. 11. Nov. 2001. pp34 - 41.

[111] B. Brumitt, B. Meyers, J. Krumm, A. Kern, and S. Shafer, “EasyLiving: Technologies for Intelligent Environments”, in the *Proceedings of the Second International Symposium on Handheld and Ubiquitous Computing*, Sept. 2000, pp97-119.

[112] G. Prochart, R. Weiss, R. Schmid, and G. Kaefer, "Fuzzy-based Support for Service Composition in Mobile Ad Hoc Networks," in *Proceedings of the IEEE International Conference on Pervasive Services*, 2007, pp. 379-384.

[113] B. Mathieu, S. Meng, and M. Kleis, “A P2P Approach for the Selection of Media Processing Modules for Service Specific Overlay Networks,” in the *Proceedings of the IEEE International Conference on Internet and Web Applications and Services*, Feb. 2006, pp. 103-108.

[114] D. Agrawal, S. Calo, K. Lee, J. Lobo, and D. Verma, Policy Technologies for Self-Managing Systems. IBM Press, 2009.

[115] A. Joshi and D. Chakraborty, "GSD: A Novel Group-based Service Discovery Protocol for MANETS," in *Proceedings of the Fourth IEEE Conference on Mobile and Wireless Communications Networks*, 2002, pp. 140-144.

[116] M. Yokoo, *Distributed Constraint Satisfaction: Foundation of Cooperation in Multi-agent Systems*. Springer, Berlin , 2001.

[117] Al-Oqily, and A. Karmouch, "Towards automating overlay network management," *Journal of Network and Computer Applications*, Vol. 32, Issue 2, March 2009, pp. 461-473.

[118] S. Jiang, Y. Xue, and D. Schmidt, " Minimum Disruption Service Composition and Recovery over Mobile Ad Hoc Networks," in *Proceedings of the Fourth Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services*, 2007, pp. 1-8.

[119] M. Perttunen, M. Jurmu, and J. Riekki, "A QoS Model for Task-Based Service Composition," in *Proceedings of the Fourth International Workshop on Managing Ubiquitous Communications and Services*, 2007, pp. 11-30.

[120] J-Sim, "Home (J-Sim Official)," *J-Sim* , January 28, 2005. [Online]. Available: http://sites.google.com/site/jsimofficial/. [Accessed: Sept. 2008].

[121] B. Corrie, H. Wong, T. Zimmerman, S. Marsh, A. Patrick, J. Singer, B. Emond, and S. Noel, "Towards Quality of Experience in Advanced Collaborative Environments," in the *Proceedings of the Third Annual Workshop on Advanced Collaborative Environments*, 2003, NRC 46521.

[122] P. Flegkas, P. Trimintzios, and G. Pavlou, "A Policy-based Quality of Service Management System for IP DiffServ networks," *IEEE Network Magazine*, Vol. 16, No. 2, March/April 2002, pp. 50–56.

[123] L. Lymberopoulos, E. Lupu, and M. Sloman, "QoS Policy Specification— a Mapping from Ponder to the IETF Policy Information Model," in the *Proceedings of the 3rd Mexican Int. Conf. comput. Sci.*, September 2001.

[124] L. Lymberopoulos, E. Lupu, and M. Sloman, "An adaptive policy based management framework for differentiated services networks," in the *Proceedings*

of the IEEE 3rd Int. Workshop Policies Distrib. Syst. Netw., June 2002, pp. 147-158.

[125] G. Valérie, D. Sandrine, K. Brigitte, D. Gladys, and H. Eric, “Policy Based Quality of Service and Security Management for Multimedia Services on IP Networks in the RTIPA Project,” in the *Proceedings of the IFIP/IEEE Int. Conf. Manage. Multimedia Netw. Serv.*, Oct. 2002, pp. 25–35.