

AMERICAN UNIVERSITY OF BEIRUT

A FRAMEWORK FOR LTE-A PROXIMITY-BASED
DEVICE-TO-DEVICE
SERVICE REGISTRATION AND DISCOVERY

by
SALAM ADNAN DOUMIATI

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Engineering
to the Department of Electrical and Computer Engineering
of the Faculty of Engineering and Architecture
at the American University of Beirut

Beirut, Lebanon
January 2015

AMERICAN UNIVERSITY OF BEIRUT

A FRAMEWORK FOR LTE-A PROXIMITY-BASED
DEVICE-TO-DEVICE
SERVICE REGISTRATION AND DISCOVERY

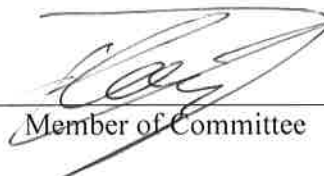
by
SALAM ADNAN DOUMIATI

Approved by:



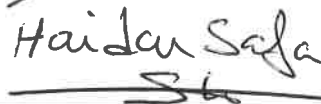
Dr. Hassan Artail, Professor
Electrical and Computer Engineering

Advisor



Dr. Zaher Dawy, Professor
Electrical and Computer Engineering

Member of Committee



Dr. Haidar Safa, Associate Professor
Department of Computer Science

Member of Committee

Date of thesis defense: January 22, 2015

AMERICAN UNIVERSITY OF BEIRUT

THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name:

Doumiati Salam Adnan
Last First Middle

Master's Thesis Master's Project Doctoral Dissertation

I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

I authorize the American University of Beirut, **three years after the date of submitting my thesis, dissertation, or project**, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

Signature

Date

ACKNOWLEDGMENTS

At this moment of accomplishment, it gives me great pleasure in expressing my gratitude to all those people who have supported me and had their contributions in making this thesis possible. First and foremost, I must acknowledge and thank The Almighty Allah for blessing, protecting and guiding me throughout this period.

I would like to express my gratitude to my supervisor Prof. Hassan Artail for the useful comments, remarks and engagement through the learning process of this master thesis. I have been extremely lucky to have a supervisor who cared so much about my work, and who responded to my questions and queries so promptly. You have been a tremendous mentor for me. One simply could not wish for a better or friendlier supervisor.

I would like to thank the rest of my thesis committee Prof. Zaher Dawy and Prof. Haidar Safa for their valuable advices and constructive criticisms. I would like also to expand my thanks to Dr. Ahmad El Hajj for his friendly assistance with various problems all the time.

Words are short to express my deep sense of gratitude towards all my friends in lifting me uphill this phase of life. Thank you Houry Seraydarian, Lama Shaer, Rasha Al Khansa, Nour Kouzayha, Adel Ejeh, Nizar El Zarif, Gilbert Badaro, Joe Akl Korban for the undying love and unconditional support you have provided me. Thank you Farah Saab, Joseph Loutfi and Khaled Bakhit for the unforgettable moments we shared in Qatar during AICCSA'14 conference. Thank you Raneen Daher, Nour Baalbaki, Karim Jamal Eddin, Zaher Masri, Mahmoud Kaissi, Ali Fakhreddine and Maxcim Yassine for believing in me that I can finish my manuscript on time.

I would especially express my warm thanks to the one who taught me to always be "*To ∞ and beyond*".

A special thanks to my family. Words cannot express how grateful I am to Mom and Dad. Your prayers for me were what sustained me thus far. I would like to pay high regards to my sister Dr. Samah, my brothers Dr. Moustapha and Dr. Hassan and my sister-in-law Dr. Nisrine for their sincere encouragement and inspiration throughout my research work.

Finally, I would like to thank my lovely little angle, my niece, HANA for being my "Porte-Bonheur".

AN ABSTRACT OF THE THESIS OF

Salam Adnan Doumiati for

Master of Engineering

Major: Electrical and Computer Engineering

Title: A Framework for LTE-A Proximity-Based Device-to-Device Service Registration and Discovery

This thesis contributes in a major way to the mainstream efforts that aim to realize the goals of LTE Advanced (LTE-A), which in turn were set to reach data rates that near 1 Gigabits per second in the downlink and 500 Megabits per second in the uplink for the future wireless communication networks. This work adapts the paradigm of Cloud Computing over the framework of Device-to-Device (D2D) proximal communications in order to offload major traffic volumes from the core network and thus enable it to grant higher data rates to mobile users. Existing approaches to the evolution of cellular network technologies have been driven by the ever-increasing need for capacity and coverage. Our proposed work introduces a platform in which mobile devices, mostly smart phones, can offer network services to other nearby devices, and thus acts as service end points (providers), thus resembling in this respect to hotspots. Hence, proximity-based D2D is accomplished while at the same time, the service provider mobile devices form transient focal points in the network, and hence act as dynamic base stations, or in Cloud Computing terminology, Cloudlets. With the Cloud Computing interface, mobile devices seeking particular services can discover providers and subsequently communicate with them directly, but with the help of the network whose role is limited to assisting in the service provider discovery process. In this way, our platform will serve to shift wireless network traffic from the core network, and thus achieve the objective of traffic offloading, but perhaps more importantly, serve the community at large by creating an environment of widespread collaboration among mobile users. This capability can introduce several positive aspects within any community, through 1) helping tourism and foreign nationals by making it simple and seamless to obtain needed services; 2) improving social ties among the members of the society; and of course 3) helping the economy through creating a more conducive environment for thriving businesses. From an implementation point of view, our solution is in line with the recommendation of 3GPP in terms of utilizing the 3GPP-proposed network elements designated for offering proximity based services, and introducing no changes to the rest of the LTE system. Our analytical and experimental results proved the viability and effectiveness of the system in helping mobile users discover needed services offered by providers who are in proximity, and therefore communicate with them directly in a peer to peer fashion.

CONTENTS

ACKNOWLEDGMENTS	v
ABSTRACT.....	vi
LIST OF ILLUSTRATIONS.....	ix
LIST OF TABLES.....	xii
LIST OF ABBREVIATIONS.....	xiii
Chapter	
I. INTRODUCTION	1
A. Background.....	2
1. D2D Concept.....	2
a. Comparison of D2D With Other Technologies	4
2. Cloud Computing Concept.....	5
B. Objectives / Significance.....	7
II. RELATED WORK.....	11
A. In the Literature.....	11
1. D2D Identification.....	11
a. Without Network Support Mode.....	12
b. Network-Assisted Mode	12
2. Architecture Enhancements.....	14
B. In the Standards.....	14

III. D2D SERVICE REGISTRATION AND DISCOVERY	19
A. System Model	19
B. System Design Requirements	22
1. ProSe-Enabled Mobile Device	22
2. Registered UE.....	23
3. Application Registered	25
a. App Registration for a UE in Consumer Mode.....	29
b. App Registration for a UE in Provider Mode.....	30
4. Service Discovery.....	34
C. Illustrating Scenario	42
IV. SERVICE DISCOVERY ANALYSIS	43
A. Performance Measures.....	43
1. Signaling Overhead	43
2. Discovery Effectiveness	71
V. EXPERIMENTAL RESULTS	83
A. Requester's Intention	83
B. Provider Discovery.....	96
1. System Environment	96
2. System Performance	98
VI. CONCLUSION AND FUTURE WORK	109
A. Conclusion	109
B. Future Work	111
BIBLIOGRAPHY	113

ILLUSTRATIONS

Figure	Page
1.1 D2D communication concept	3
1.2 D2D pair	9
3.1 Overview of the LTE-A network with the ProSe functional components.....	20
3.2 Control plane for PC3 interface [35]	23
3.3 ProSe Function discovery	24
3.4 Control plane for PC4a interface [35].....	24
3.5 UE registration.....	25
3.6 Downloading Apps by the user equipment from a Third Party Application Server.	26
3.7 Envisioned application user interface for an SaaS application.....	28
3.8 Application registration: consumer mode.....	29
3.9 ProSe apps on mobile phone.....	30
3.10 Proposed database of Application Server (case of SaaS Application)	31
3.11 Application Registration: provider mode	32
3.12 High level service discovery system (matching on keywords level).....	35
3.13 PC2 interface [35].....	37
3.14 PC4b interface [35].....	39
3.15 Proposed service discovery system.....	40
3.16 Service discovery in [35]	41
4.1 Threshold distance	43
4.2 UE ₁ moving $D_{max}/2$ distance	44
4.3 Diameter header content [40]	46
4.4 AVP header content [40]	47

4.5 Description of a point as two coordinates [43]	48
4.6 “Location-Estimate” information element content [43]	48
4.7 Shape description of a point [43]	49
4.8 Providers distribution.....	64
4.9 Number of requesters’ effect on the traffic (a) inner traffic (b) wireless medium traffic.....	67
4.10 Request’s rate effect on traffic (a) inner traffic (b) wireless medium traffic.....	68
4.11 Number of Matching ProSe Functions’s effect on the traffic (a) inner traffic (b) wireless medium traffic	69
4.12 (a) Traffic for a pedestrian requester (b) Traffic for a requester driving a vehicle.	70
4.13 Score of each keyword per service versus the number of concatenated services ...	74
4.14 Services’ popularities and providers’ choices	75
4.15 Service matching between requester and provider	76
4.16 Keywords matching between requester and provider.....	77
4.17 Score per keyword per provider versus the number of intended services	79
4.18 Requester changing directions to reach provider pm_h	80
4.19 Number of providers discovered in proximity versus the length of the area.....	82
4.20 Sequence diagram of the entire system design	82
5.1 Similarity matrix	85
5.2 (a) Normalized similarity matrix (b) Upper triangle diagonal of the matrix	87
5.3 (a) Similarity Number Matrix (b) Upper triangle diagonal of the matrix.....	88
5.4 Concatenated list.....	89
5.5 Requester’s choice from the concatenated service keywords list.....	90
5.6 Average Number of hits and false positives versus minimum number of necessary common keywords to put in one pool (a) 3 keywords/request (b) 5 keywords/request.	91

5.7 Average number of hits and false positives versus the number of requesters' keywords.....	93
5.8 Average number of hits and false positives versus the number of requesters' keywords for different threshold values	95
5.9 D2D throughput as a function of the D2D link distance [23].....	97
5.10 Providers' and requesters' areas	98
5.11 Provider's chosen service and keywords	99
5.12 Requester's request creation	99
5.13 List of providers matching keywords	100
5.14 (a) List of providers matching keywords along with their matching indices and distances (b) Filtered list.....	101
5.15 Average number of discovered providers versus the percentage number of providers in the network (regardless of proximity)	103
5.16 Average number of discovered providers versus the number of providers' keywords (regardless of proximity).....	104
5.17 Average number of discovered providers versus the number of requesters' keywords (regardless of proximity).....	105
5.18 Average number of discovered providers (per request) in requester's proximity versus the length of the simulation's area.....	106
5.19 (a) the average number and (b) the standard deviation of discovered providers versus the requester's speed (Pedestrian scenario, $D_{\max}=25\text{m}$).	107
5.20 (a) the average number and (b) the standard deviation of discovered providers versus the requester's speed (Vehicle scenario, $D_{\max}=25\text{m}$).	108

TABLES

Table	Page
1.1 Comparison between D2D and other technologies.....	5
2.1 ProSe Standards	18
3.1 IDs held by each entity	33
4.1 Diameter header size.....	46
4.2 AVP header size.....	47
4.3 Coding of type shape [43].....	49
4.4 Diameter protocol information elements [40]	50
4.4 Diameter protocol information elements [40] (Continued)	51
4.4 Diameter protocol information elements [40] (Continued)	52
4.5 ProSe standards information elements.....	53
4.6 Newly created AVPs.....	54
4.7 Packet's size.....	63
4.8 Parameters used in the equations	65
4.9 Packets' size and number of times they are sent (inner traffic).....	66
4.10 Packets' size and number of times they are sent (wireless medium traffic).....	66
5.1 Services and associated information.....	84
5.2 Default parameters	97

ABBREVIATIONS

3GPP	Third-Generation Partnership Project, 1
LIPA	Local IP Access, 1
SIPTO	Selected IP Traffic Offload, 1
IP	Internet Protocol, 1
eNodeB	Evolved Node B, 1
QoS	Quality of service, 2
ProSe	Proximity-Based Services, 2
D2D	Device-to-Device, 2
UE	User Equipments, 2
LTE-A	Long Term Evolution-Advanced, 3
EPC	Evolved Packet Core, 3
E-UTRAN	Evolved Universal Terrestrial Radio Access Network, 3
WLAN	Wireless Local Area Network, 4
STaaS	Storage as a Service, 5
PaaS	Platform as a Service, 5
NaaS	Network as a Service, 5
IaaS	Infrastructure as a Service, 5
DaaS	Data as a Service, 5
M2M	Machine-to-Machine, 8
_P	Provider, 9
_R	Requester, 9
App ID_X	Application Identification, 9
SAE	System Architecture Evolution, 11
OFDMA	Orthogonal Frequency Division Multiple Access, 12
SIP	Session Initiation Protocol, 13
URI	Uniform Resource Indicator, 13
NAS	Non-Access Stratum, 13
PSCF	Proximity Service Control Function, 14
P-GW	Packet Data Network GateWay, 14
PDN	Packet Data Network, 14

MME	Mobility Management Entity, 14
TSG	Technical Specifications Group, 14
SA	Service and System Aspects, 14
RAN	Radio Access Network, 14
WG	Working Group, 15
TR	Technical Report, 15
TS	Technical Specs, 15
E-UTRA	Evolved UMTS Terrestrial Radio Access, 15
PLMN	Public Land Mobile Network, 16
EPS	Evolved Packet System, 16
HSS	Home Subscriber Server, 20
SLP	Secure User Plane Location Platform, 20
IMS	IP Multimedia Core Network Subsystem, 21
SP	Service Provider, 21
SR	Service Requester, 21
SLP	Secure User Plane Location Platform, 21
SUPL	Secure User Plane, 21
U-plane	User-plane, 21
OMA	Open Mobile Alliance, 21
HPLMN	Home Public Land Mobile Network, 22
APIs	Application Programming Interfaces, 22
FQDN	Fully Qualified Domain Name, 23
DNS	Domain Name Service Function, 23
IMSI	International Mobile Subscriber Identity, 24
SIM	Subscriber Identity Module, 24
EPUID	EPC ProSe Subscriber ID, 26
App Server	Application Server, 26
ALUID	Application Layer User ID, 26
PFID	ProSe Function ID, 27
Guid	Globally Unique Identifier, 31
TTL	Time to Live, 33
LCS	Location Services, 38

MLP	Mobile Location Protocol, 38
D_{\max}	Maximum distance separating two UEs defined to be in proximity, 43
D_T	Threshold distance to update the location, 44
ε	Distance corresponding to the time elapsed between a UE sending a location update until the core network reacts, 44
AVP	Attribute-Value Pair, 46
GAD	Universal Geographical Area Description, 49
NAI	Network Access Identifier, 50
PIR	ProSe-Subscriber-Information-Request, 55
PIA	ProSe-Subscriber-Information-Answer, 56
REQ	Message Request, 57
L	Total Number of PLMNs, 63
l	Number of PLMNs containing providers having matched keywords with the request, 63
ω	Number of Keywords matching applications at one provider, 63
U	Total number of UEs, 64
v	Average Speed, 64
T_m	Time to move $D_{\max}/2$ meters, 64
T_r	Request Period, 64
N_r	Number of Request in T_m per UE, 64
RWP	Random Waypoint Mobility Model, 72
Q	Total Number of Requesters Keywords, 72
Y	Requester's Set of Keywords, 72
y_q	One of the Q requester's keywords ($Y = \{y_1, y_2, \dots, y_Q\}$), 72
M	Total Number of Concatenated Lists, 72
Cl	Set of Concatenated Lists, 72
cl_m	One of the M concatenated lists ($Cl = \{cl_1, cl_2, \dots, cl_M\}$), 72
$ cl_m $	Length of cl_m , 72
$f_{y_q}^{cl_m}$	Term Frequency of Keyword y_q in cl_m , 72
f_{y_q}	Number of Concatenated Lists that have y_q in it, 72
Ct	Intended Concatenated List, 73
C	Total Number of Services in Ct, 73

ct_c	One of the C services in t ($Ct = \{ct_1, ct_2, \dots ct_C\}$), 73
$ ct_c $	Number of Keywords in ct_c , 73
$f_{y_q}^{ct_c}$	Term Frequency of Keyword y_q in ct_c , 73
f_{y_q}	Number of Services that Contain Keyword y_q , 73
SR	Matching Services between Request and Concatenated Services, 73
R	Total Number of SR, 73
P	Total Number of Providers, 75
PV	Set of Providers, 75
pv_p	One provider ($PV = \{pv_1, pv_2, \dots pv_P\}$), 75
SP	Set of Providers' Services, 75
sp_p	Corresponding Service to each Provider, 75
W	Total Number of Keywords Chosen by a Provider, 75
K	Provider's Keyword, 75
K^{sp_p}	Provider's keyword K from sp_p 's keywords set, 75
H	Number of Providers Having Matching services, 75
SM	Set of Matching Services between Requester and Provider, 75
a	Length of the Area, 80
A'	Subarea in the simulation area A, 80
$f_{XY}(x,y)$	Spatial Node Distribution, 80
LD	Levenshtein Distance, 84
s_k, s_l	Services k and l, 85
$g_{k,l}$	Weight of Similarity between two Services, 85
T	Threshold That Defines the Requester's Knowledge, 89
std	Standard Deviation, 90
t	Time of Sending the Request, 99
dist	Euclidean Distance between Requester and Provider, 100

CHAPTER I

INTRODUCTION

The usage of smart phones, tablets and various new applications throughout the world has exploded during the recent years throughout the world and will continue to increase exponentially according to the Wireless World Research Forum who envisions in 2020 Seven Trillion wireless devices serving Seven Billion people [1]. This growth will lead to huge mobile data traffic on the network. According to Cisco predictions in [2], the global mobile data traffic will outgrow global fixed data traffic by three times, reaching 10.8 exabytes per month (1 exabyte equals 1018 bytes), or an annual rate of 130 exabytes, by 2016. Thus, the flag has been raised to find ways in order to increase network capacity and accommodate the bandwidth consuming applications and services.

The most straightforward solution is to improve the capacity of cellular networks by adding new base stations, but this is very expensive for the operators. Therefore, The Third-Generation Partnership Project (3GPP) has defined data offloading as an alternative solution to cope with this problem. 3GPP Rel-10 has been working on two key data offloading areas: Local IP Access (LIPA) and Selected IP Traffic Offload (SIPTO) [3]. LIPA allows a direct communication between an IP-enabled mobile terminal and a local network where both are connected to the same base station (eNodeB). SIPTO, on the other hand, offloads selective IP traffic to the Internet at home or in enterprise environments.

However, there are two main shortcomings in these two methods:

- They only relieve core network congestion, not radio congestion, since the data-offload points are positioned at or above eNodeBs, and not at the mobile terminals.
- They do not maintain the quality of service (QoS) for relevant applications that use the cellular network [4].

Accordingly, the 3GPP SA1 (services) working group has been studying since 2011 a new Rel-12 item, named Study on Proximity-Based Services (FS_ProSe), targeting the potential requirements for an operator to integrate Device-to-Device (D2D) communication in their network [5]. This technology has been proposed as a promising concept for improving user experiences and resource utilization in cellular networks by taking advantage of users' proximity.

A. Background

1. *D2D Concept*

In order to appreciate the usefulness of D2D, it is worth reminding how regular conventional cellular communications work. Normally, two mobile devices (e.g., smart phones), referred to as User Equipments (UEs) communicate via a radio uplink to the Base Station (known as eNodeB), via a core network uplink and then downlink, and finally through a radio downlink, even if the two UEs are right next to each other.

On the other hand, in D2D, mobile devices in proximity of each other can establish a direct local link and bypass the base station or access point, but after coordinating with the core network via the eNodeB. This type of communication is

referred to as local offloading or local source-sink application. The difference between conventional cellular communication and D2D can be illustrated in Figure 1.1.

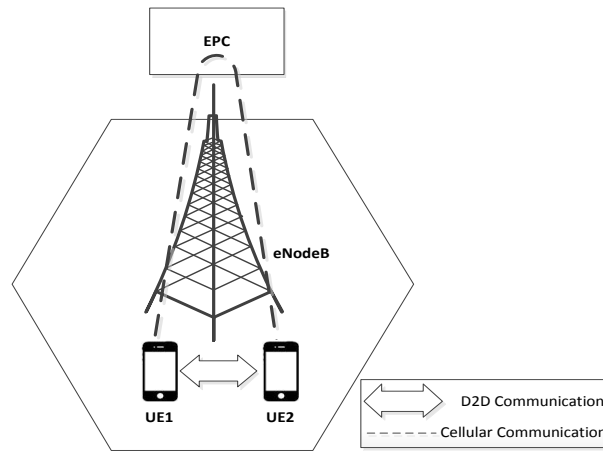


Figure 1.1 D2D communication concept

Hence, device-to-device communication promises several gains in cellular networks, in addition to offloading data:

- It enables very high bit rates, low delays and low power consumption [6].
- It improves spectrum reuse and system throughput since the radio resources may be simultaneously used by cellular and D2D links.
- It offers a hop gain since the link in the D2D mode is single rather than using both an uplink and downlink resource [4].

In this section, we provide a brief overview of the architecture of the Long Term Evolution-Advanced (LTE-A) cellular network, showing the main entities. This will help in reading the thesis, given that we make reference to the entities that make up the network. The core network of the LTE system, also known as Evolved Packet Core (EPC), is responsible for the overall control of user equipment and establishment of the bearers (connections) to the devices via the Evolved Universal Terrestrial Radio Access Network (E-UTRAN) consisting of eNodeBs (Base Stations).

a. Comparison of D2D With Other Technologies

D2D present many advantages over the existing technologies used for peer-to-peer communication like Wireless Local Area Network (WLAN) [7] and Bluetooth [8]:

- These two aforementioned technologies require the user intervention: by doing manual device pairing in Bluetooth and by entering user-defined settings for the access points in WLAN [9]. However, in D2D, the operation is fairly transparent to the user.
- The access of these technologies is uncertain in terms of their availability: Bluetooth has a star topology which may prevent a new user to be connected if the number of slaves overcomes the predefined number. As for WLAN, there is a condition of channel freedom so that users can transmit. This uncertainty may annoy the users who will stay in a trial-and-error process to access the channel [9]. However, users in D2D communication are allocated resources by the infrastructure in a guaranteed way.
- Although such technologies can operate without any infrastructure assistance, they lack of node synchronization and assisting security procedures which can be offered by the cellular network in case of D2D communication [10].
- Bluetooth and WLAN do not assure the Quality of Service QoS of the ongoing session since it depends on the channel conditions. However, in D2D communication, the QoS of the radio bearer between the D2D pairs is assigned, controlled and maintained by the cellular network [11].
- D2D communication is a source of income for wireless operators other than Bluetooth and WLAN which work independently [11].

The comparison between these different technologies can be summarized in the following table:

Table 1.1 Comparison between D2D and other technologies

	Bluetooth	WLAN	D2D
User Intervention	✓	✓	✗
Availability	✗	✗	✓
Synchronization	✗	✗	✓
Security	✗	✗	✓
QoS	✗	✗	✓
Source of Income	✗	✗	✓

2. *Cloud Computing Concept*

By relying on sharing resources to achieve coherence, Cloud Computing is intensively progressing and reaching new levels by giving permission to Internet users for sharing infrastructures, platforms, and software provided by the cloud. Globally, Apple with its iCloud, Google with its Drive, Amazon with its Cloud Drive, Microsoft with its Skydrive, Dropbox and many others offer Storage as a Service (STaaS). Also, Google with App Engine, Amazon with AWS Elastic Beanstalk, Microsoft with Windows Azure Compute and Heroku also offer platform as a service (PaaS), and these are only two types of available cloud computing services. Many other types of services are also available: Network as a Service (NaaS), Infrastructure as a Service (IaaS), Data as a Service (DaaS), etc. This trend does not only affect ICT companies offering services to consumers, as it also affects companies having their own private smaller clouds (referred to as cloudlets) to offer services, data, and storage to their employees and users [12]. Cloudlets, “smaller clouds” are based on powerful computers that usually serve nearby users and excel at offloading content and tasks from mobile devices [13-15].

It is a popular belief that mobile devices are not powerful enough to run compute-intensive tasks and applications. However, very recently, researchers are starting to realize that mobile phones, like other computing devices, also follow Moore's Law, and have seen a huge leap forward in terms of CPU speed and memory capacity, even more than other categories of devices in recent years. Thus, it has become increasingly possible for mobile devices to rely on themselves or on other nearby devices for obtaining network services. In this regard, researchers at AT&T Labs [16] proposed to use mobile devices as mobile cloudlets. When a certain mobile device needs to execute a task, it can either do it using its own resources or use resources of a nearby mobile device by delegating tasks when in need. The proposed framework relies on broadcast messages for discovering mobile cloudlets. As was elaborated, the real difficulty of using mobile devices as cloudlets is the fact that they are mobile, and can become out of range without prior notice and prevent the tasks that were already started from being successfully completed. The work in [16] suggests that the discovery phase be repeated periodically or in response to specific events such as moving or walking. Nevertheless, the study in [16] does not describe a communication protocol for supporting and maintaining the formation of cloudlets.

The issue of device capabilities is not the only concern, as although 4G networks such as LTE and LTE-A have very high efficient physical and MAC layer paradigms, they are still suffering from the increasing loads on the backhaul due to the increasing demand for services and for reducing end-user latency. Hence, new and disruptive network paradigms are needed to improve the end-user experience in terms of offered services, reduced service latency, increased throughput, while keeping the overall overhead minimal on the network backhaul. A byproduct of such paradigms is

offloading large amounts of traffic from the core network, thus contributing to the objectives of 3GPP. One of these new paradigms is the D2D communication, which is represented as a promising component in the next generation. D2D is defined as a direct communication between two mobile devices without or with minimal intervention from the base stations or the core network of the cellular network. Eventually, D2D is the new trending topic of the cellular networks for the operators.

B. Objectives / Significance

The aim of this research is to develop a comprehensive D2D communication for data under the support of the existing LTE-based cellular system. We seek to accomplish this through the design of a system that works within the framework of the existing LTE network entities with new added functions to support proximity services.

The key functions of D2D communications include service registration, peer discovery, D2D bearer establishment, and switching the path to D2D data offloading. In our work, we will mostly focus on the first two steps, meaning service registration and peer discovery, as they represent the bulk of the work to implement D2D.

The topic of device-to-device communication gained a lot of attention from researchers who proposed solutions for the following challenges:

- Resource allocation (e.g. spectrum and energy) between cellular communication and ad hoc D2D communication users, and resource management to coordinate interference by using power control [17] and by using multi-antenna transmission techniques [18].
- Deciding on whether users should communicate in D2D or in cellular mode, referred to as proper mode selection [19]. Researchers have assumed that D2D

communication setup is already supported in LTE-A, but this assumption is not totally valid since existing LTE-A proposed architectures and protocols are not ready yet to support D2D.

Few works have been done to propose enhancements at the network architecture level in order to integrate D2D communication in LTE-A. The main contribution of our work is going beyond the high level architecture to develop a detailed design for handling D2D *data* traffic, such that the impact on the current design (entities, communication protocols, functions and roles, packets design) is minimized. That is, one of our objectives is to generate a design that can be implemented with minimal changes to the existing LTE infrastructure, by exploring the existing functionalities of the elements of the LTE-A architecture.

The other potential contribution is to extend the concept of network services adopted in the LTE-A literature and the use cases proposed in the ProSe standards. On one hand, the D2D technology is used in the literature in the context of multicasting [20], [21], peer-to-peer communication [22], video dissemination [23-25], Machine-to-Machine (M2M) communication [26], and cellular offloading [27]. On another hand, the ProSe use cases presented in the 3GPP study item [5] are categorized under commercial or social use, network offloading and public safety. Some scenarios are presented in this study item but it mainly focuses on public safety where E-UTRAN coverage is absent. We believe that users having ProSe-enabled devices and being in proximity can benefit from this situation in a broader context: we propose for the services offered to be any mobile application, commercial or personal, that runs on a mobile device in response to the request of others for the purpose of benefiting these other devices, and is ProSe compliant. For this, our goal is to design a system that

would allow providers of arbitrary services, who are subscribers to the cellular system, to advertise their services through the LTE-A system, and other users (also subscribers) to discover such services and consume them by bringing the cloudlets concept to the D2D framework. An example would be a new restaurant owner advertising the food services her restaurant offers, like food orders, customizable meals, reservation, delivery options, etc.

Figure 1.2 shows a pair of devices communicating through D2D technology. It illustrates two users (UE_R and UE_P) running the same application on their devices (with application identification App ID_X, a unique identifier characterizing this particular application) which can run in consumer mode or provider mode.

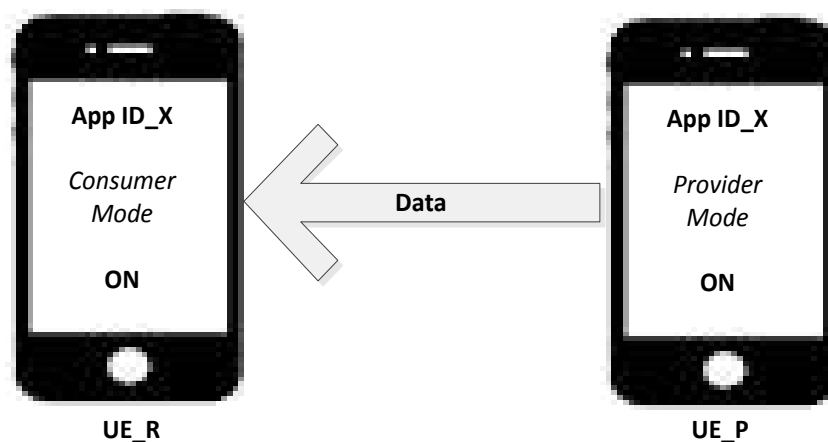


Figure 1.2 D2D pair

The mobile app, in need of a certain service (UE_R), runs in consumer mode and asks the app on the other device holding the resources (running in provider mode UE_P) to provide it with. We seek to accomplish this through designing a system that works within the framework of the existing LTE network entities trying to make minimal changes to it.

The significance of this work is that it will use the existing entities without adding new major ones. To that end, we suggest some modifications in the exchanged messages between the user equipment (UE) and the Evolved Packet Core (EPC), propose some additional functions to the entities, and one table to the Application Server. For this reason, we take [5] as a baseline for our discovery scheme, and more specifically the Evolved Packet Core (EPC)-level discovery, where the core network is involved in determining the proximity of the devices.

In the rest of this report, Chapter II discusses related work and reveals the contributions of the proposed system, which we describe in Chapter III. Chapter IV provides an analytical analysis of the system's average performance. Chapter V presents the experimental results and discusses their significance. Chapter VI finishes the thesis with concluding remarks and suggestions for future works.

CHAPTER II

RELATED WORK

This chapter is divided into two main parts where the first section presents the work done so far by the researchers in the literature while the second part details the related work achieved in the standards.

A. In the Literature

This section reviews the proposed schemes for Device-to-Device (D2D) identification followed by the suggested enhancements for the protocol architecture and System Architecture Evolution (SAE) in order to incorporate D2D communication in the LTE-A.

1. *D2D Identification*

Peer discovery has a similar functionality as that of cell search in LTE by which the UE determines the time and frequency parameters that are necessary to demodulate the downlink and determine the cell identity [28]. In addition to time and frequency, devices should also meet in space. The discovery of devices willing to participate in a D2D communication is a challenge in terms of energy consumption, since scanning for devices may end up draining the device battery [29]. The literature on this topic is divided between relying on UE's abilities in discovering other mobile devices or integrating the network core in this process.

a. Without Network Support Mode

Without any coordination with the network, the discovery process can be made possible via some procedure, but it would be time and energy consuming. Some authors [29] base their discovery process proposal on the transmission of beacons between the devices using Orthogonal Frequency Division Multiple Access (OFDMA) and building on the existing beacon design of the 3GPP Long Term Evolution (LTE). To resolve the problem of synchronization when multiplexed together in the same OFDMA symbols, the devices are divided into groups that use different patterns to transmit in different beaconing opportunities.

A new mobile communication system is introduced in [30] to realize a new form of proximity-aware networking. This system is founded on an implementation of “wireless sense” named FlashLinQ, which allows devices to discover each other and communicate directly. In their design, the authors kept the involvement of the network at a minimum, mainly to provide synchronization signals to devices. FlashLinQ is the base for a new technology named as LTE Direct (invented by Qualcomm [31]) integrated in the 3GPP standard [32] that studies the architecture enhancements to support Proximity-based Services (ProSe).

b. Network-Assisted Mode

Other researchers adopted schemes that benefit from network assistance. Two mechanisms for D2D communication session setup and management are proposed: through detecting D2D traffic, or by using dedicated System Architecture Evolution (SAE) signaling [9]. In the first option, the potential D2D traffic is earmarked on the fly by the gateway after processing the IP headers of the data packets and tunnel headers.

Although this method works for any peer-to-peer IP traffic without service differentiation, it adds overhead to the network. Concerning the second option, by using dedicated signaling, a D2D Session Initiation Protocol (SIP) session request can be separated from a generic SIP session request: UE1 calls UE2 using a SIP invite message with a specific address format, where the well-known SIP Uniform Resource Indicator (URI) of UE2 is specified, followed by an extension indicating the preference for a local session. The SIP invite message, encapsulated in a Non-Access Stratum (NAS) control message, is received by a light SIP handler added to the Mobility Management Entity (MME). With this approach, the NAS messages corresponding to D2D will be processed by the handler, while the ordinary ones are handled by the MME. The advantage of using dedicated signaling is that it does not require a SIP server in the Internet, thus leading to faster session setup. Reported simulation results showed an increase of throughput up to 65 percent for a network with D2D communication, when compared to an ordinary cellular network. It is noteworthy to mention here that the work in [9] does not discuss the peer discovery process and does not explain how UE1 can know the URI of UE2.

The identification of D2D traffic by an existing or added architectural LTE entity has been an active research subject. For example, some authors [10] present two alternatives for detecting D2D candidates in a network assisted scheme, differentiated by whether the detection takes place before or after the start of the D2D session. In the a-priori scheme, the role of the network can be expanded or reduced: the eNodeB can only broadcast the assignment of beacon resources so that the server and the client find each other, or the eNodeB can work as a mediator between them by redirecting the request of the client to the D2D server (pre-registered), so that the latter generates the

beacon. However, in the a-posteriori scheme, the eNodeB identifies the D2D pairs either by a token agreed on by the two devices, or by analyzing the source and destination IP addresses.

2. Architecture Enhancements

The authors in [4] proposed to add a Proximity Service Control Function (PSCF) to the Packet Data Network GateWay (P-GW), which is the gateway that terminates the interface towards the Packet Data Network (PDN), which in turn refers mostly to the Internet. The addition of PSCF was for detecting the presence of D2D traffic flow and allocating a pair identity to the communicating UEs.

On the other hand, other authors [33] introduce a new logical entity called the D2D server, separated from the other existing entities (but interfacing with them), that is responsible for device identifier allocation, policy management, assistance in location determination, call establishment, UE capability tracking, service support, and mobility tracking. They also propose enhancements to the Mobility Management Entity (MME) so that it supports new D2D related information during the attach procedure, and therefore be able to identify the devices' D2D service capabilities.

B. In the Standards

D2D communication gained a lot of attention from the 3rd Generation Partnership Project (3GPP) which conducted an intensive work at the Technical Specifications Groups (TSGs) level (Service and System Aspects (SA) that describes the service requirements and the overall architecture of the 3GPP system, and Radio Access Network (RAN) that studies the radio aspects) and their corresponding Working

Groups (WGs). A series of documents were developed in Release 12 under the title of Proximity Services (ProSe). These services are defined in [5] as services that can be provided by the 3GPP system based on UEs being in proximity to each other. The standardization work for D2D followed the 3GPP development stages: It started by a “Study Item” that delivered a “Technical Report” (TR) [5] describing the use cases for Proximity Services from the user point of view.

A “Work Item” was then divided into three stages, each for a specific purpose: Stage 1 for studying service aspects; Stage 2 for describing technical realization on the architecture level to integrate these services; and Stage 3 for detailing protocols implementing the architecture in Stage 2, and also defining security aspects. It delivered several “Technical Reports” (TR) [32], [34] that led to “Technical Specs” (TS) [35], [36], [37]. Another group was working on the radio aspects for D2D technology to define evaluation models [38]. In the following, an overview of each “Technical Report” and “Technical Specs” is presented.

In [5], a feasibility study for Proximity Services (ProSe) is presented. This study identifies the ProSe key features consisting of ProSe discovery and ProSe communication. The discovery process occurs when the user equipment (UE) announces its identity using the LTE air interface (Evolved UMTS Terrestrial Radio Access, E-UTRA) to another UE which will recognize that it is in its vicinity. This announcement makes the discovery open or close, depending on whether it needs an explicit permission from UE or not. As for ProSe communication, it describes the communication path established between two UEs making it direct (direct mode), or routed via the local eNodeB (i.e., locally routed). Moreover, this study presents the services that can benefit from two users having ProSe-enabled devices and being in

proximity to each other by analyzing different use cases (social networking applications, public safety, parking services, etc.) and scenarios (e.g., subscribers from different Public Land Mobile Networks (PLMNs), roaming subscribers). It should be mentioned here that this study mainly focuses on public safety usage within and outside the network coverage. Furthermore, the functional requirements for the operators are also indicated in the context of integrating this technology in their networks and monitoring it to provide users a seamless switch from the user plane communication path through the Evolved Packet Core (EPC) to a ProSe E-UTRA communication path and vice versa. Finally, this study highlights the charging and security requirements for users using ProSe.

The ProSe concept introduced in [5] led to updates in [36] and [37] by adding special normative specifications for ProSe. The document [36] initially described the service requirements for the Evolved Packet System (EPS) that comprises the Evolved Packet Core (E-UTRA) and the evolved radio access network (E-UTRAN) to maintain its characteristics in terms of latency, user data rates, system capacity, coverage, and operational costs. As for [37], it initially described the service aspects of charging and billing for the 3GPP system, but as additional services were introduced, new charging requirements were added for the use of both ProSe features: Discovery and Communication. Knowing that D2D technology may add traffic to the system, new requirements were considered. Definitions for ProSe Discovery and Communication were extracted from [5] along with the requirements for Proximity Services and for public safety, and added to [37].

Considering the above, the TRs and TSs [5], [36], [37] describing the service aspects led to a technical realization stage, namely Stage 2. In this TR [32], many

solutions are proposed to enhance the existing architecture and integrate ProSe functionality, while at the same time presenting the key issues that should be considered and evaluating their impacts on the existing network. The solutions related to ProSe discovery are mainly divided into two types: EPC-level discovery and direct discovery. In the first type, the network acts as a mediator between the two UEs, detects their proximity and notifies them about it. The other type is a direct one, where UEs recognize by themselves their neighboring UEs using the LTE air interface. This TR describes a high level architecture for each solution showing the entities needed and the interfaces between them, along with the other 3GPP existing entities, plus the required functions to make a PLMN support ProSe.

Some of the solutions presented in the above TR phase were documented in normative specifications in the new technical specification [35], mainly related to the EPC-level discovery. This TS mentions the roles of the newly added entities and the interfaces between them. Moreover, it describes the information flow procedure for UE registration, application registration, UE proximity request to the network, UE location reporting to the EPC, and proximity alert from the network to the concerned UEs (steps derived from the previous TR [32]). Although, this technical specification defines the protocol stacks on the control and user planes for each interface, it does not describe them in details. This was left for future work in the next stage, meaning Stage 3.

The ProSe features were also studied from the security perspective since ProSe pose threats for the user's privacy, delivering a technical report [34] that studied the security requirements in Stage 3. This subject is however out of the scope of the subject of this thesis. On the other hand, other groups, like the Radio Access Network (RAN) group, worked on the radio aspects and ended up by writing a technical report [38],

where evaluation models for channel, traffic, and mobility were presented. It also defined the performance evaluation metrics for discovery and communication.

Nevertheless, the work of 3GPP is still at the concept level, as it is missing key design elements, and needs to be generalized to suit real-world scenarios. For example, how a service provider develops and deploys his offered service application for other network users to discover and interact with is not yet clear.

We conclude this subsection with Table 2.1, which summarizes the standards that are related to ProSe.

Table 2.1 ProSe Standards

Technical Specs Group	Description	Stage	Specs Number	Goal
Group Services and System Aspects	Service Aspects	Stage1 (Study Item)	TR 22.803 [5]	To study use cases and service requirements
		Stage 1 (Work Item)	TS 22.278 [36]	To specify service requirements for the Evolved Packet System
			TS 22.115 [37]	To specify service requirements (charging and billing)
	Technical Realization	Stage 2	TR 23.703 [32]	To define the architectural enhancements
			TS 23.303 [35]	
	Security Aspects	Stage 3	TR 33.833 [34]	To study the threats and the security requirements
Group RAN		RAN	TR 36.843 [38]	To define the evaluation models (channel, traffic, mobility)

CHAPTER III

D2D SERVICE REGISTRATION AND DISCOVERY

In this chapter, we provide the complete design of D2D service registration and discovery architecture and describe the interaction between the different components of the system. Our work extends the approach described in the standards [38], by removing the requirement for the requesting device to know the ID of the providing device, and more importantly, laying out the grounds for overlaying a mobile computing framework over the capability of proximity-based device-to-device communications. In this section, we describe our system model before discussing our proposed framework in the section that follows.

A. System Model

The two most involved processes of D2D communications are service advertisement and service discovery. In our proposed system, service registration and discovery involve the network in the discovery process, thus avoiding the time and energy consumption issues associated with discovering D2D candidates without network support, as in [29], [30]. Our proposed architecture also adopts the a-priori scheme [10] where D2D pairs are detected before the start of the D2D session. Moreover, we keep the involvement of the core network to a minimum: mainly for a Service Provider (SP) to register a service and for a Service Requester (SR) asking for a service to discover the SP is in its proximity.

Below, we detail the various components of the proposed system and their interactions with one another. The two main steps of D2D data communication are

service registration and service discovery. We base our design on the Evolved Packet Core (EPC)-level discovery mentioned in [32] where it is the responsibility of the network to determine the proximity of the user equipment and inform them about it. “Proximity”, as defined in [5], takes different criteria for discovery and communication: for discovery, the criteria include radio range and geographic range as for communication, it include others like: range, channel conditions, achievable Quality of Service (QoS). Since our work focuses on ProSe discovery, we mean by “Proximity” a geographical range i.e. two UEs in near distance to each other.

The high level architecture of our design is illustrated in Figure 3.1.

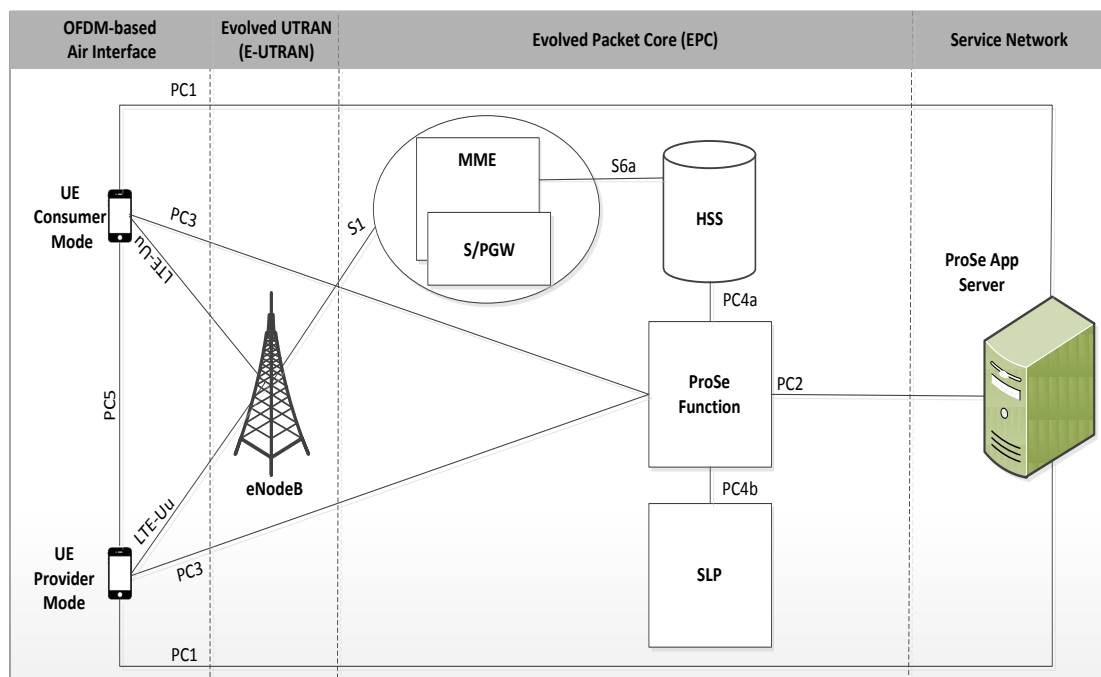


Figure 3.1 Overview of the LTE-A network with the ProSe functional components

Figure 3.1, which is based on [35], shows the already existing entities in the LTE-A network: base stations (eNodeB), Mobility Management Entity (MME), Serving Gateway (S-GW), Packet Data Network GateWay (P-GW), Home Subscriber Server (HSS), Secure User Plane Location Platform (SLP), with the additional logical function

ProSe Function. This last element turns the Public Land Mobile Network (PLMN) into a network that supports Proximity Services. The D2D UE pair (one in consumer mode and the other in provider mode) communicates with the core network via the eNodeB as part of the connection setup. The core entities involved in the discovery process are all the existing ones in the figure except the S/P-GW as they connect UEs to the external network, like the IP Multimedia Core Network Subsystem (IMS), which is not our case. The key functional elements for this process are described below:

- The MME is responsible for all mobility related functions (tracking and paging). In our framework, it also caches a copy of the user's ProSe profile after being authenticated by the HSS, and informs the eNodeB about the user's permission.
- The HSS is a data repository for subscribers' profiles that authenticates/authorizes user access to the system, and more specifically will check whether the requesting users are ProSe subscribers or not.
- The SLP can be a server residing in the network or a network equipment stack. It obtains location information for the UE using Secure User Plane (SUPL) which is supposed to be the user-plane (U-plane) location technology developed by OMA (Open Mobile Alliance [39]) for positioning over wireless network, based on secure user plane IP tunnels.
- The ProSe Function generates the IDs of the ProSe users after being authorized by the HSS and handles these IDs along with their corresponding application layer user IDs. It also stores a list of authorized applications IDs to use EPC-level ProSe discovery. Moreover, the ProSe Function plays the role of location services client (SLP agent) to communicate with the SLP and be aware of the UEs' locations to determine their proximity. It should be noted that according to

[35] there is only one ProSe Function per Home Public Land Mobile Network (HPLMN).

- The ProSe Application Server contains the applications offering services developed using Application Programming Interfaces (APIs) for ProSe which are provided by the 3GPP operator during the service agreement. It is the entity on the service network from which the user downloads the apps. It also stores the identities of the ProSe users, as defined at the network level, and maps these identities to the application layer user identities which identify specific users within an application. Moreover, the ProSe function ID corresponding to each user is also saved there.

Note that the above entity roles are based on [35]. Further functions added to these entities will be discussed in the next section.

B. System Design Requirements

A mobile user willing to participate in a D2D communication in order to benefit from Proximity-based Services (ProSe) should fulfill the following criteria, as is discussed next.

1. *ProSe-Enabled Mobile Device*

A user must be first equipped with an LTE-A mobile device that can communicate with the Core Network via the corresponding interfaces. Moreover, this device should have the capability to run ProSe applications on it, meaning the applications having the ProSe capability features: the ability to discover, to be discovered, and to communicate with the discovered devices.

2. Registered UE

This ProSe enabled device should be subscribed to an operator service in order to be authorized to run ProSe enabled applications on it. The registration of the device occurs in the Home Public Land Mobile Network (HPLMN) where the subscriber's profile is held in a logical function, named ProSe Function which makes this PLMN a network that supports Proximity Services. It is assumed that each PLMN contains only one logical ProSe Function. The ProSe Control signaling for the registration process of the user device occurs over PC3, a reference point between the ProSe Function and the UE, which relies on Evolved-Packet Core (EPC) user plane for transport, meaning over Internet Protocol (IP), as it is shown in Figure 3.2.

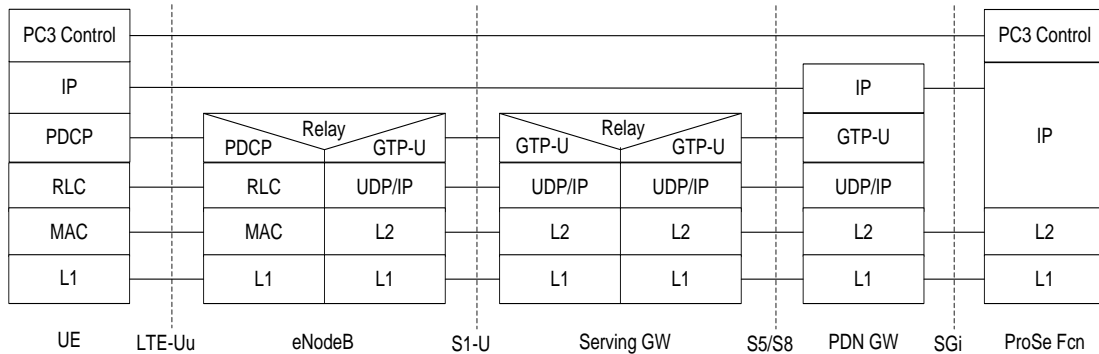


Figure 3.2 Control plane for PC3 interface [35]

To allow for this IP communication, the user should be aware of the IP address of the ProSe Function, that's why he needs to start a ProSe function discovery. From the HPLMN ID broadcasted by the HPLMN, the UE constructs a Fully Qualified Domain Name (FQDN) that uniquely identifies a ProSe Function and interacts with the Domain Name Service Function (DNS) to translate this FQDN and get the corresponding IP address of the ProSe Function. By this, the ProSe Function ID is known to the UE. The ProSe Function discovery is depicted in Figure 3.3.

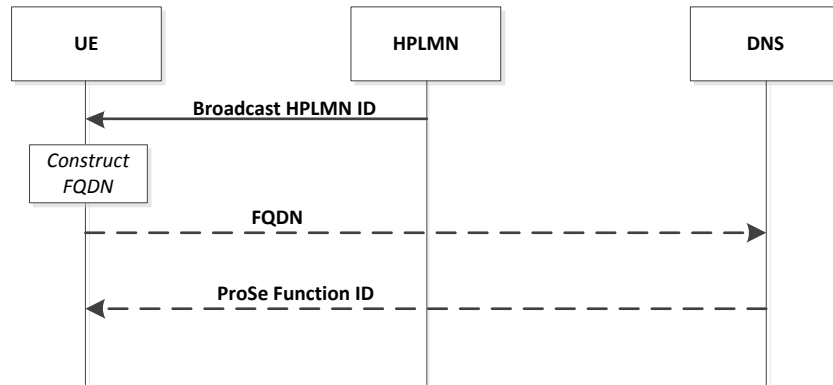


Figure 3.3 ProSe Function discovery

To register the device on the ProSe Function, the user identifies his UE to the network by sending, over PC3, his International Mobile Subscriber Identity (IMSI - the unique number associated with each mobile phone and stored in the Subscriber Identity Module, SIM). The ProSe function authenticates him by checking with the Home Subscriber Server (HSS) if he or she is allowed to use ProSe features that consist of ProSe discovery and ProSe communication for this device. This information about authentication/authorization access is exchanged between the HSS and ProSe Function over the PC4a interface by using Diameter Protocol as shown in Figure 3.4.

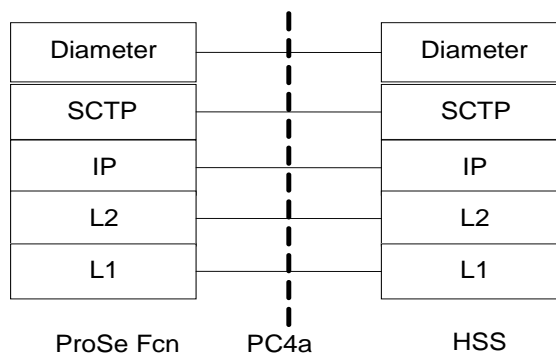


Figure 3.4 Control plane for PC4a interface [35]

After authenticating the UE, the ProSe Function creates an EPC ProSe Subscriber ID (EPUID) for the registered device and sends this ID back to the UE in

order to be used later during the discovery process as it is illustrated in Figure 3.5.

UE , in this figure, can be either a Requester or a Provider.

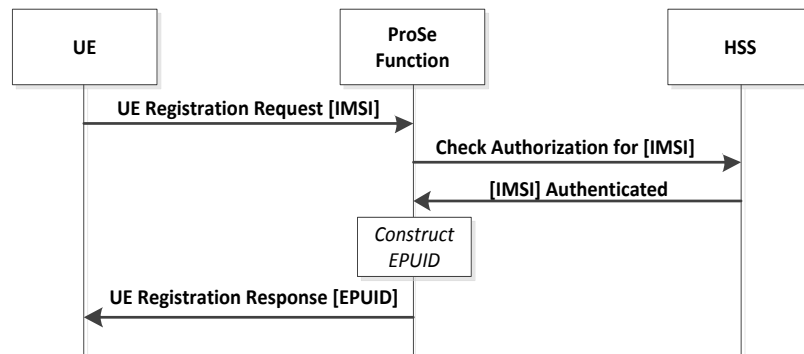


Figure 3.5 UE registration

By this, the UE is registered as ProSe subscriber and ready to run applications that support Proximity Services on it, named as ProSe enabled applications.

3. *Application Registered*

The user equipped with a ProSe-enabled device and registered in the network as a ProSe subscriber, can download a ProSe application (offering services to other ProSe-enabled devices) from an application server through an operator application distributor or Application Store. This server may be operated by the 3GPP network operator or by a third-party service provider. In the latter case, the developers should sign a contract with the 3GPP operator in order to authenticate their application server and get an Application Developer ID as well as an Application ID which globally defines this specific application and identifies their 3rd party App Server platform. Moreover, in order to make these apps having the ProSe capability features, meaning the ability to discover, be discoverable and to communicate with the discovered device, the third-party service provider should build the applications based on Application Programming

Interfaces (APIs) provided by the 3GPP operator in the service agreement as shown in Figure 3.6.

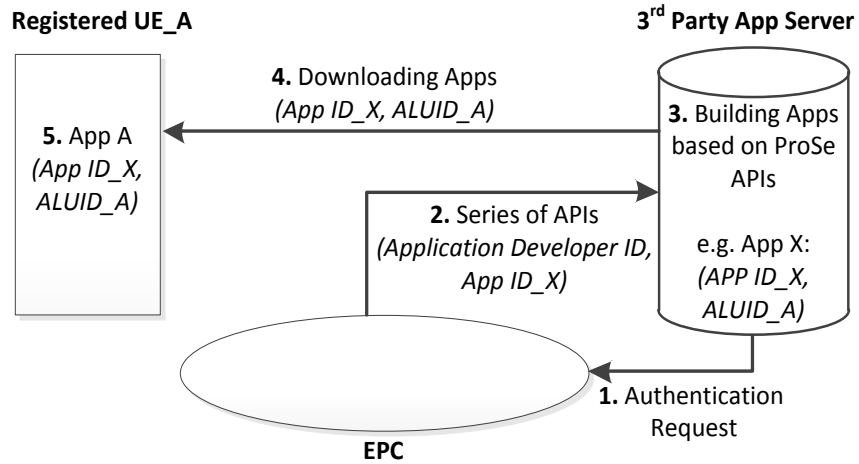


Figure 3.6 Downloading Apps by the user equipment from a Third Party Application Server

The communication between the user and the Application Server occurs over PC1, the reference point connecting the application instance to the server. The Application Server assigns him an Application Layer User ID (ALUID) to identify him within the context of this specific application as it is illustrated in Figure 3.6. Note that the Application Registration is the same for the requester as well as for the provider. UE_A represents either case.

To activate the ProSe features on these applications, the user should also authenticate and authorize them through the ProSe Function that caches a list of all the IDs of the applications allowed to use ProSe features. Note that the authentication is done on a per-application basis. The user identifies his authorized device to the network by his EPC ProSe Subscriber ID (EPUID) and sends it to the ProSe Function along with the ID of the application (Application ID) he wishes to authorize and his own ID on this application (ALUID). This message containing all these IDs is sent through PC3, the

reference point connecting the UE to the ProSe function which checks that the user as well as the application requested is authorized. Once authenticated, the ProSe Function forwards this message to the App Server adding to it its own ID (the ProSe Function ID (PFID)) in order to indicate that this user has requested to use ProSe for that application. The App Server stores the IDs (ALUID, EPUID and PFID) to map between the application IDs and the users' ID in the discovery step and sends a ProSe Registration Response message to the ProSe Function as a sign of success for the application registration. To terminate this process, the ProSe Function replies to the UE by an acknowledgment containing the authorized discovery range class for this application. According to [5], the discovery range class can be short, medium or maximum based on geographical distance or radio conditions. This range defines how far a UE holding this app can discover another radio signal or can be discoverable. The user has the freedom to choose one of these allowed ranges while requesting to communicate with a nearby device holding this app.

In our design, we propose to develop the applications on the application server to run in two modes: consumer (default) mode and provider mode. Some devices may be willing to lend their extra resources in terms of software, data, or network to other mobile devices, playing the role of mobile cloudlets. By this, the D2D pair will be a UE in the consumer mode and the other in provider mode (mobile cloudlet). An app runs in consumer mode when the user is requesting services, and runs in provider mode when the user is willing to share his device's resources. After downloading the app to his device, the user can choose whether he will allow the app to run in provider mode through a configuration interface, which could look like the prototype in Figure 3.7.

As shown in the figure, in the first screen, the user who wants to be a provider for a certain service turns the “Provider Mode” on and fills the corresponding information.



Figure 3.7 Envisioned application user interface for an SaaS application

- Discovery type: open or restricted

According to [32], the open discovery type lets the mobile device be found by another UE, different than the restricted one where the UE can only be seen by a certain group of people. In our design, the restricted type can be linked to a group ID in a social network chosen by the user. As its name shows, the drop down list of the “Social Network Name” contains the names of the social networks to which a user belongs: “Facebook”, “Twitter”...

- Application name

The names of the ProSe applications existing on the mobile device are linked in a way to this user app interface in order to appear in the drop down list corresponding to this field. The user can add (+) or remove (–) applications to apply the provider mode rules.

- Availability

The user defines the time during which he or she will be available as a provider for this service using the digital clock gear.

Once the information are filled, the “Next” button leads the user to the second page named as “Application List & Keywords Association” where the names of the apps chosen in page 1 will appear. Once checked, the user can add (by using the + sign) and remove (by using the – sign) the keywords corresponding to each app.

a. App Registration for a UE in Consumer Mode

If the UE is running on consumer mode, no additional messages for application registration are exchanged between the mobile device and the application since the app is running in its default mode. The steps followed by the UE are as mentioned earlier in Section 3.2.3 and can be summarized by the following sequence diagram.

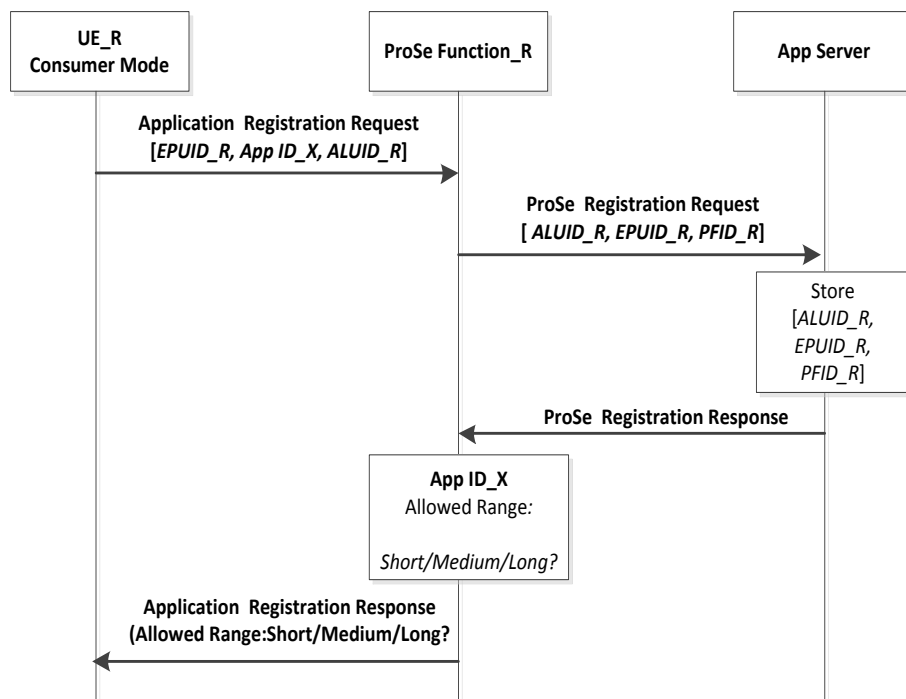


Figure 3.8 Application registration: consumer mode

b. App Registration for a UE in Provider Mode

If the UE specifies provider mode, it should satisfy certain criteria to prove its capability to serve a mobile cloudlet, restricting by this the providers' number by the network. Depending on which of the three intended mobile cloudlet service types the device will offer (SaaS, DaaS, and NaaS), the criteria will be different since in our design we assume that each application corresponds to a certain type of service meaning to say an application for Data as a Service is different than the one for Software as a Service but may exist on the same server. For example, in the case of SaaS, the device is expected to host software apps that are to be run under the supervision of the downloaded ProSe App (provider) as possible services to same-type consumer apps. In case of DaaS, the device is supposed to host the necessary data (e.g., song files) to be sent to consumer apps. Finally, for NaaS, the device must have, for example, an active WiFi connection to the Internet (e.g., via subscription or privilege) and can configure itself as a hotspot for requesting devices running the same-type consumer app to connect through to the Internet. We note that the same UE can host different apps each one corresponding to a certain type: SaaS, DaaS and NaaS as depicted in Figure 3.9.

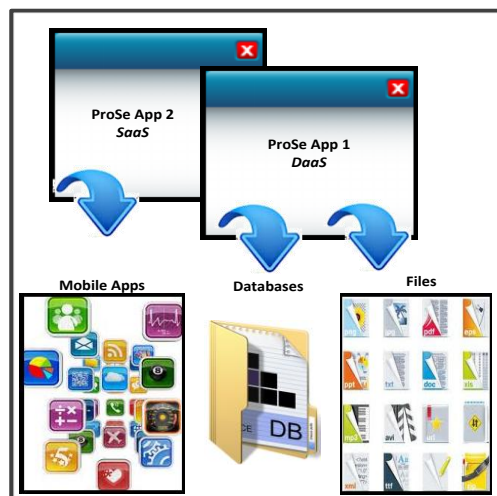


Figure 3.9 ProSe apps on mobile phone

After choosing the provider mode from the application user interface, the UE should prove its capabilities to the Application Server. In the case of Data as a Service (DaaS), for instance, the user, through an interface, could configure the local SQLite database to store the names and other metadata of available video and image files that are to be shared with nearby LTE-A users. If the UE does not fulfill the required criteria to be accepted as a provider, the provider mode requested will be rejected.

Otherwise, if the App server proves the user’s competency to be in provider mode, it sets the Boolean value for “Provider” to “True” in the “Subscriber” table present in its schematic database that is shown in Figure 3.10. The “Applications” and “Subscriber” tables are connected by a one-to-many relation since each application has many subscribers who downloaded it; they are related by the Application ID (App ID) playing the role of the key between them. This key is of type a Globally Unique Identifier (guid) that identifies a particular application within all the applications existing on this server.

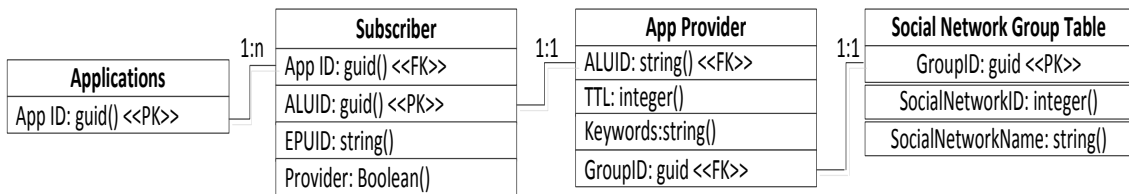


Figure 3.10 Proposed database of Application Server (case of SaaS Application)

The “Subscriber” table contains fields related to the user’s profile: the Application Layer User ID (ALUID) which is the user’s ID on the application server, the EPC ProSe Subscriber ID (EPUID) of string type is the user’s ID in the network and the Provider field which is of Boolean type that is checked when the application is running on provider mode. The “Subscriber” and “App Provider” tables are connected by a one-to-one relation via the Application Layer User ID (ALUID) which plays the

role of the key between them. The other fields existing in the “App Provider” table will be discussed later in the context of their usage. We note that this database corresponds to Software as a Service Application, and so, the database of the other applications will only differ by the attributes appearing in the “App Provider” table.

The UE working in provider mode should register the services he or she offers in the network as well as in the App server in order to be discoverable by other UEs as it is shown in Figure 3.11. When the user checks the “Provider mode” field; he will be asked if he wants to be discoverable by anyone holding this application (open discovery) or only by restricted people (restricted discovery). The application could be linked to social network where the user can choose a group of people to discover him or her (or the user can precise a group ID).

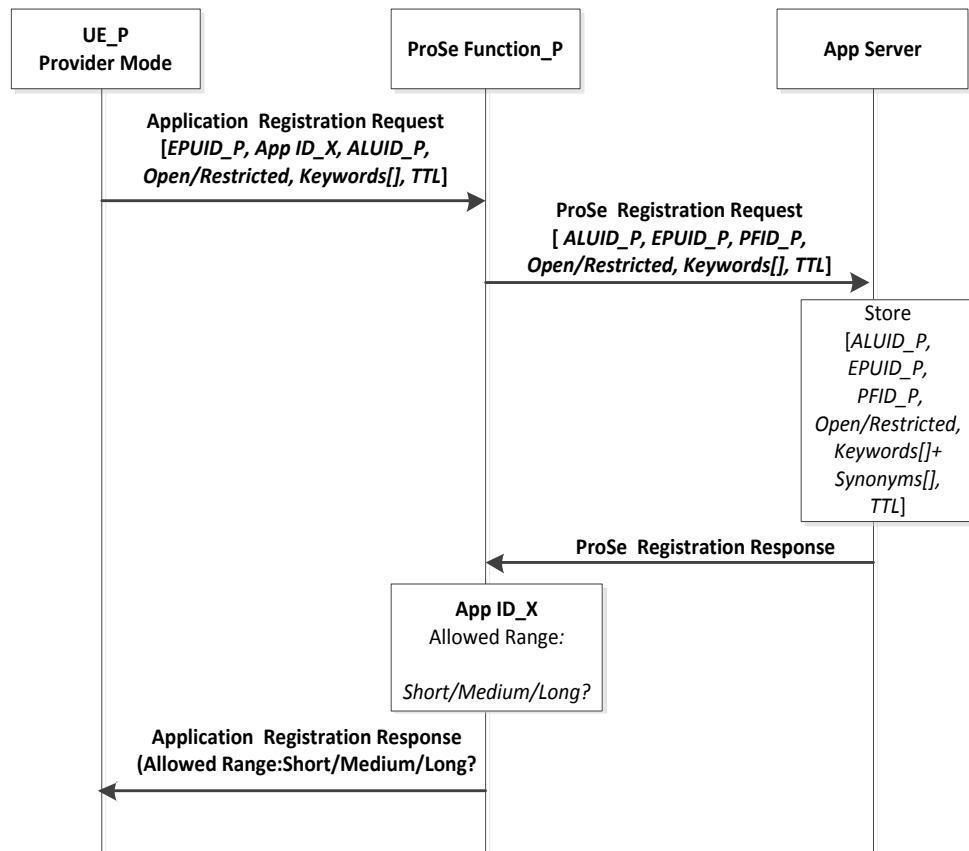


Figure 3.11 Application Registration: provider mode

For instance, a professor wants to share an excel file containing the grades only for his students. Note that the user should also precise a Time to Live (TTL) which indicates the duration of his availability as a provider. Once TTL expires, the “Provider” field will be set to “false”. In addition to this, the user should mention some keywords that help in discovering its offered service for other users running their applications in consumer mode. These keywords will be concatenated with a semi colon and sent to the App Server which has an additional function to find all its corresponding synonyms.

Now, the application server, playing the role of a directory, saves all these information related to this user cloudlet in its database. Now that the user’s device as well as the ProSe application running on it is authorized, the user can start sending proximity requests to the network. The records for each entity are summarized in Table 3.1. The application server holds the association between the IDs at the application level (ALUID) and the network level (EPUID). We should point out that nothing precludes a device running in provider mode to also run in consumer mode by requesting a same-type service, but different in the specific offerings.

Table 3.1 IDs held by each entity

At the About	UE	ProSe Function	App Server
User	IMSI	IMSI	
	EPC ProSe Subscriber ID (EPUID)	EPUID	EPUID
Application	Application Layer User ID (ALUID)	ALUID of both UEs (until request ends)	ALUID
	Application ID		Application ID
ProSe Function		ProSe Function ID (PFID)	PFID

4. *Service Discovery*

A key function in the discussed framework is service discovery. Part of this functionality is identifying the general D2D service type, which is implicitly provider through same-type ProSe-compliant applications (services). That is, a device wishing to discover devices running an app (service) must already be a subscriber to this service in order to run it as a client that knows how to communicate with the provider. However, discovery has to be more specific. For example, an SaaS cloudlet could offer parking information services, while another SaaS cloudlet may offer Mexican food ordering services. It follows that a ProSe-compliant SaaS app interested in finding free parking spots in nearby parking garages should only communicate with SaaS provider apps running on nearby devices that offer parking information.

To realize the above capability, when a device subscribes to a ProSe application as a provider, it should supply to the network (application server) a set of keywords. With this setup, a user wanting a particular proximal service will supply search keywords that are compared against the registered keywords (and their synonyms) to determine the appropriate cloudlets, and present them to the user. Obviously, multiple matches could occur, in which case, the user is free to select which one to connect to.

As it is shown in Figure 3.12, a phone in provider mode downloads from the Application Server many ProSe Apps of the SaaS and DaaS types, where the first app communicates with the mobile apps running on the user equipment while the other one has access to databases and files present on the phone. The application server acts as a directory by saving users' information related to the type of the application downloaded on their UEs as well as to the keywords defining this app. We show the profiles of two users BBB and CCC are saved in the App server's database and both users are in the

requester's vicinity. Note that the phone (provider mode) illustrated in this figure can be user BBB or CCC.

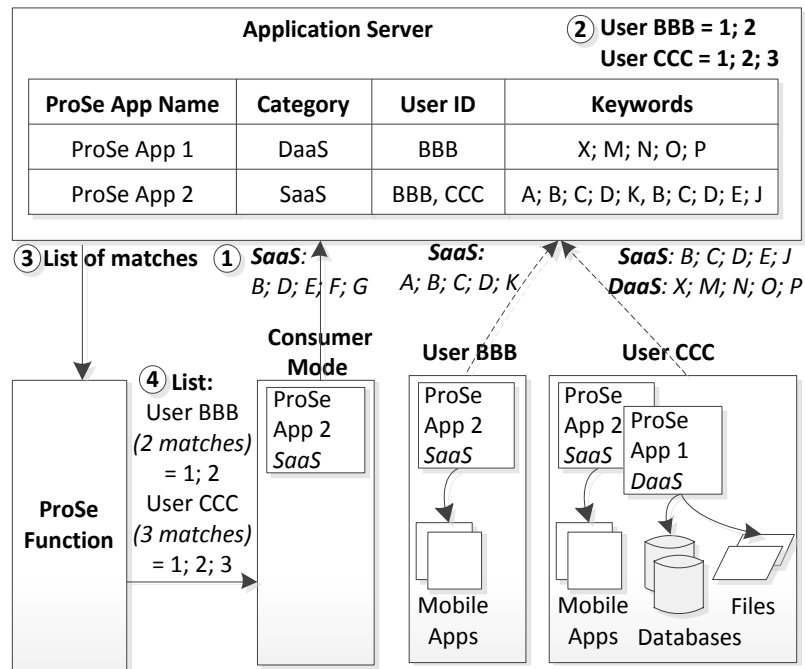


Figure 3.12 High level service discovery system (matching on keywords level)

User BBB's UE holds a DaaS app and an SaaS app, where the first one is defined by keywords X, M, N, O while the second by A, B, C, D. Assuming a phone in consumer mode, holding ProSe App 2 (SaaS) is searching for an application using the keywords B, D, E, F, G (step 1). The Application Server detects 2 matches with the registered keywords for user BBB and 3 matches for user CCC (step 2). It responds back by sending the number of matches and indices of the matched keywords, along with the provider ID to the consumer phone to choose which provider it wants to contact. If user CCC, with the higher number of matches is chosen, the consumer will inform the ProSe Function about this (step 4) in order to establish a bearer between him and the provider.

In more technical terms, a device sends a proximity request to the network through messages that ask the ProSe Function to find nearby targeted devices, or to alert it when other devices come around. The user defines the proximity criteria when he selects a range class for this app. That is, a user who has chosen “short” range class (e.g., short class corresponds to 10 m) will not be informed about a UE 20 m away.

In [35], when UE_A is interested in finding UE_B, it contacts the network in order to be alerted when this device comes to its vicinity. It is therefore assumed that UE_A knows the Application User ID of UE_B. In our proposed work, we remove this assumption by making UE_A search for an Application ID along with identifying keywords for the desired *service*. The importance of keywords usage here is that they narrow down the list of providers having the same application. For instance, if UE_R is interested in finding a UE running an application that offers Mexican restaurant food services and wants to be alerted by the network when such UE is around or comes to its vicinity. If UE_R searches only by the ID of the application (i.e., without any keywords), it will get a list of all providers offering restaurant services (i.e., all types of cuisines), which may be a rather long list, thus annoying the requesting user.

Technically, since the discovery is EPC-level based, UE_R needs to contact the ProSe Function of his HPLMN to help it in finding these cloudlets. For this purpose, all UE_R has to do is to send a proximity request message to the ProSe Function, identifying itself by its EPC ProSe Subscriber ID (EPUID_R), its ID in the application, (ALUID_R), the Application ID and one or more keywords describing the service it is searching for. Moreover, UE_R should precise in this message the range class of the application he or she wants and provide the ProSe Function with its current location in order to be used for proximity calculations. Note that the request should also be defined

by a certain time (a time window). The request will thus be defined by EPUID_R, Application ID, ALUID_R, time window, range, UE_R's location, and the keyword(s). Since ProSe Function_R is not aware of the EPC ProSe IDs of UEs in provider mode for this application nor of the ID of the ProSe Function they are subscribed to, ProSe Function_R to contact the Application Server for translating ALUID_R into subscriber IDs. This is done over PC2, the interface between the ProSe Function_R and the Application Server, as depicted in Figure 3.13.

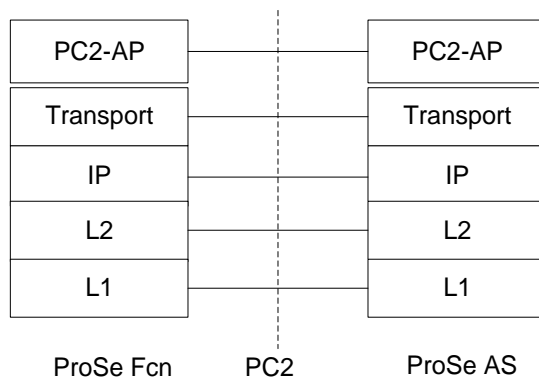


Figure 3.13 PC2 interface [35]

ProSe Function_R sends the Application ID, the application layer user ID of the requester (ALUID_R) and the keywords to the App Server. We should note that ALUID will be stored inside ProSe Function until the time window expires. The application server searches for the cloudlets that are defined under this Application ID as providers (subscribers with the “Provider” field in the “Subscriber” table set to True) and searches through their registered keywords to find a match with the keywords supplied by UE_R.

Since the consumer and the provider may be registered with different ProSe Functions, ProSe Function_R needs to know the IDs of the other ProSe Functions. The server responds back by sending all the matching EPUIDs with the keywords requested, their corresponding ProSe function Ids and their available resources to ProSe

Function_R. The latter saves this information temporarily and contacts each ProSe Function found in the list by sending a Proximity Request containing the EPUID to the corresponding ProSe Function. The ProSe Functions of the providers use EPUID to retrieve the profile of the UEs and contact the HSS to get their last known locations. As we know, the users' locations saved in the HSS database are on tracking area level, which is why each ProSe Function will compare the last known location of the provider to the requester's location and checks if they are likely to meet or no (different tracking area) doing by this a rough filtering. If they are in the same area, the provider's ProSe Function (ProSe Function_P) sends an acknowledgment to ProSe Function_R. Then, knowing the involved ProSe functions, ProSe Function_R asks each ProSe Function to start the location reporting from their corresponding Secure User Plane Location Platform (SLP). The Secure User Plane Location Platform (SLP) is a new entity that is specified in [35], whose responsibility is to keep track of the UE's locations by sending updates of registered UEs' locations (periodically, or in response to triggering events) to the corresponding ProSe Functions (playing the role of Location Services (LCS) client) over PC4b through the Mobile Location Protocol (MLP) shown in Figure 3.14.

For instance, SLP_R, keeping track of UE_R's location, updates ProSe Function_R about UE_R's location by sending a Location Services (LCS) report. In our case, ProSe Function_R will have to collect the location updates from ProSe Function_P, provided by SLP_P since UE_R has initiated the discovery. After being detected in proximity, each ProSe Function cancels the location reporting from their corresponding SLPs

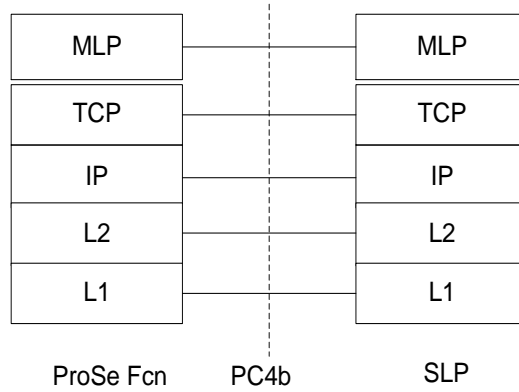


Figure 3.14 PC4b interface [35]

Finally, ProSe Function_R creates a reply packet to the UE_R containing a list of the ALUIDs each with its corresponding EPUID, sorted by their distances from the UE (nearest to farthest) and the count as well as the indices of the matched keywords. After compromising between keywords and position, UE_R chooses a certain cloudlet and informs the network about its choice by sending a Decision Message containing the ALUIDs and EPUIDs of both the requester and the provider. The Prose Function_R only keeps the information related to the chosen cloudlet and deletes the remaining ones. EPUID known, the Prose Function maps it with the corresponding ProSe Function ID. Assume the UE chosen as cloudlet is UE_P. Thus completing the discovery stage. If no providers are in range, the network responds to UE_R's request by sending an empty list prompting it that it will alert it when a provider enters its range.

The sequence diagram of discovery is depicted in Figure 3.15.

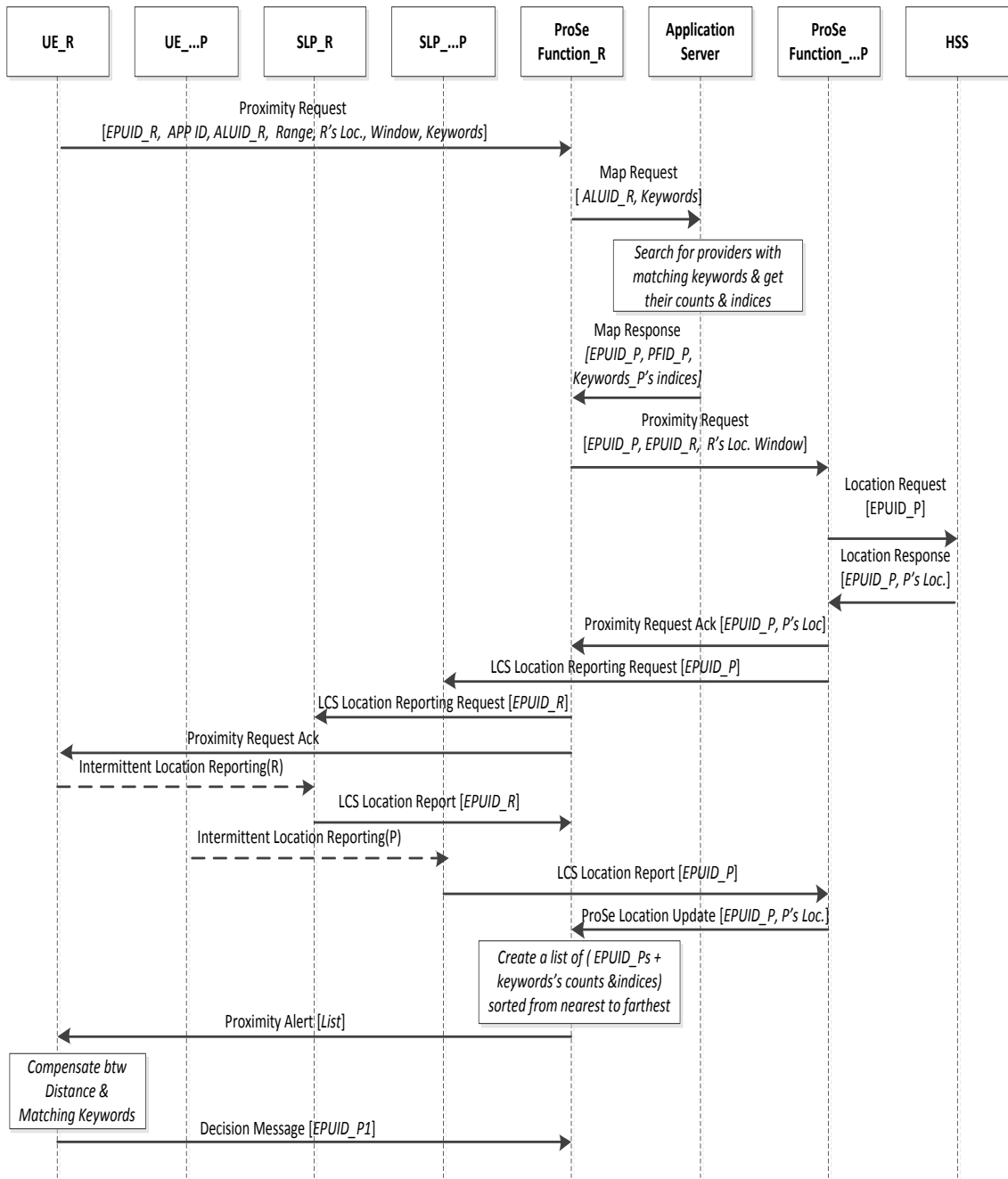


Figure 3.15 Proposed service discovery system

By comparing our proposed discovery system in Figure 3.15 to the one mentioned in the standards [35] and illustrated in Figure 3.16, we see the difference in terms of generalizing the search request by removing the assumption that the requester knows the application layer user id of the targeted UE and by integrating the metadata search option.

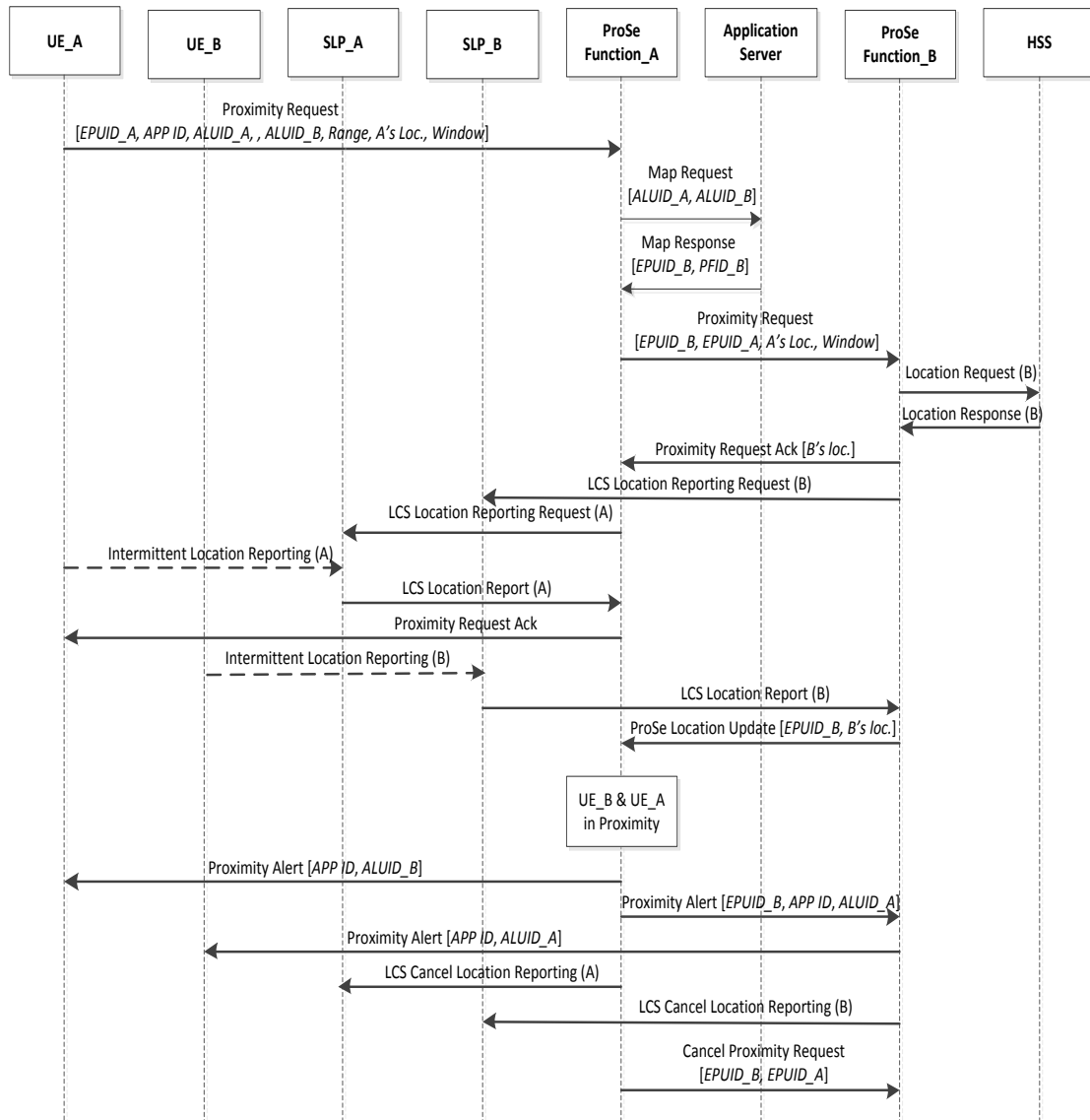


Figure 3.16 Service discovery in [35]

C. Illustrating Scenario

At the end of this chapter, we present an illustrating scenario corresponding to an application offering Software as a Service (SaaS), relating to available parking lots in a busy downtown of a major city. We suppose that Mary, Peter and John are mobile users holding UE ProSe enabled devices, registered in the ProSe Function of the same operator network as ProSe subscribers, and running on their UEs an authorized ProSe app named “FindAPark”, downloaded from an application server and also registered in the network. Peter and John’s devices periodically receive parking spot availability information from their respective lot database, which in turn gets updates from sensors installed in the lot. When Mary enters the downtown searching for a place to park her car, she uses her “FindAPark” to ask the network to help her find nearby UEs providing parking info. Moreover, she indicates in her request the discovery range class she wants for this app (e.g., “short” range class) and some keywords (e.g. short-term parking, low rate). According to Mary’s location and preferences, the network identifies Peter for being in her proximity and offering matching services and sends his application layer user ID in the list of the response message to Mary. Although John is a provider, but he is not within Mary’s “short” range, which is why his ID was not part of the list. Once Mary is aware of Peter’s ID, they can start their D2D communication, where Mary may see a map of the available parking spots, select one, and pay securely using her credit card.

CHAPTER IV

SERVICE DISCOVERY ANALYSIS

In this chapter, we present our simulation results to gain insights into the performance aspects of our proposed discover system. The performance metrics considered are as follows: 1) signaling overhead: the signaling messages added to the existing system in the standard [35] 2) the effect of the number of keywords chosen by the requesters and providers as well as others parameters' impact such as: number of providers in the network, area of simulation's dimensions and distance separating the requester and provider.

A. Performance Measures

1. *Signaling Overhead*

EPC-level discovery can create signaling in the network for activating and maintaining location reporting from the SLP since it keeps track of the user's location. In our system, we consider that a UE updates its location to the SLP only when it moves a significant distance. To analyze this threshold, we start by defining D_{max} where D_{max} corresponds to the maximum distance separating two UEs defined to be in proximity. Supposing three UEs in Figure 4.1, where UE_1 is far away from UE_2 by D_{max} meters and UE_3 is very close to UE_2. The average distance by this would be $D_{max}/2$.

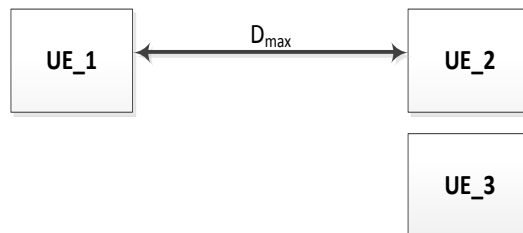


Figure 4.1 Threshold distance

Hence, the UE would update its location to its corresponding SLP only when it moves a distance D_T defined by:

$$D_T = \frac{D_{max}}{2} - \varepsilon$$

where ε is the distance corresponding to the time elapsed between when a UE sends a location update until the core network reacts. As it is illustrated below, when UE_1 goes far from UE_2 by D_T from its initial position (dotted square), it sends a location report to its SLP informing it about its new location.

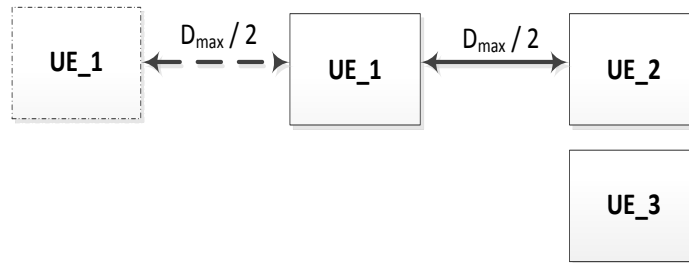


Figure 4.2 UE_1 moving $D_{max}/2$ distance

The signaling overhead depends on the dynamics of the cell, i.e. the traffic that flows between entities and the mobility of the mobile providers. In a static scenario, where nodes do not change their positions frequently and communication sessions are set-up for a longer period, the signaling overhead is limited. In a very dynamic scenario, sessions can be disconnected and switched into normal cellular communication session. In another case, if the requester is insisting on being involved in D2D sessions even after being disconnected, he should reinitiate his request to the prose function in order to find a new provider in its vicinity, by this all the steps aforementioned will be repeated leading to an increase in the network traffic.

For simulation purposes, we study different scenarios:

- UEs registered to the same Public Land Mobile Network (PLMN)
- UEs registered to different PLMN

These cases were particularly chosen knowing that the signaling load arises across inter-PLMN interfaces since the prose function of the requester should communicate with each provider's PLMN in order to retrieve the user's profile and location. However, in both cases, we consider dynamic nodes to imitate real case scenarios where mobile users can be pedestrians or riding vehicles changing frequently by this their positions. This change will cause more traffic since users need to keep updating the SLPs about their current locations.

In this part, we assume that the UEs participating in this discovery are already registered in the network and the applications running on these mobile phones are also registered on the Application Server. Our purpose in this section is to analyze the signaling overhead in the service discovery step. Assume that there is no caching in the ProSe Function. In case caching exists, the number of sent messages exchanged should be multiplied by $(1 - \text{caching hit rate})$.

We start by defining the information elements that may exist in each packet and the type of encoding used along with a brief description. The messages are exchanged using the diameter protocol [40]. All diameter messages start by a diameter header as defined in [35] where further explanation about each field can be found. The content of this header is illustrated in Figure 4.3. Our goal is to calculate the total size of this header by summing up the size of each field and is found to be equal to 160 bits as it is shown in Table 4.1.

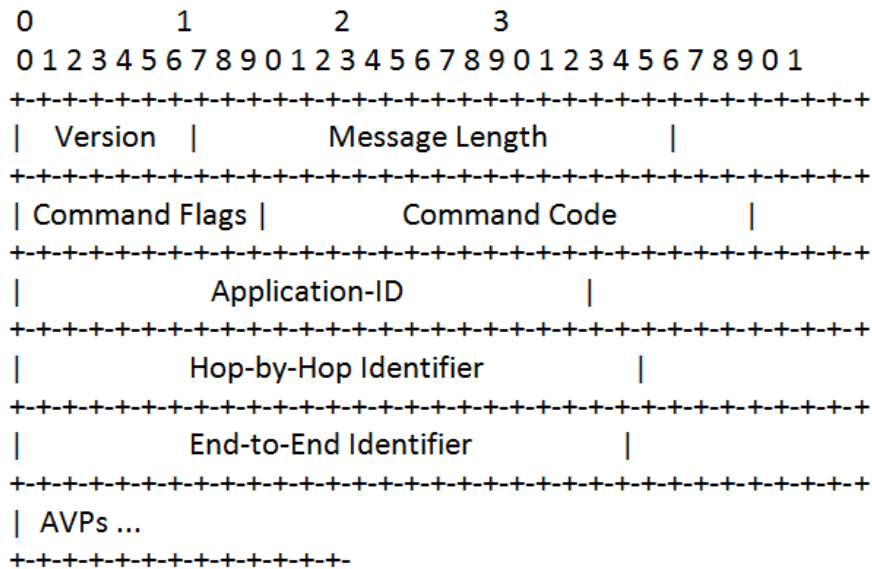


Figure 4.3 Diameter header content [40]

Table 4.1 Diameter header size

Fields	Size (bits)
Version	8
Message Length	24
Command Flags	8
Command Code	24
App ID	32
Hop-by-Hop Identifier	32
End-to-End Identifier	32
Total Size	160

As Figure 4.3 shows, the diameter header is followed by a series of AVPs where AVP is defined to be Attribute-Value Pair which is the basic unit inside the Diameter message that carries the Data (Authentication Data, Security Data, Data pertaining to Application etc). Each AVP has its own header followed by the corresponding data. For this reason, it is necessary to compute the AVP header size (which is the same for all AVPs) in order to be used in calculating the size of each AVP. The AVP header is shown in Figure 4.4 as defined in [40]. Further information about each field in this header can be found there.

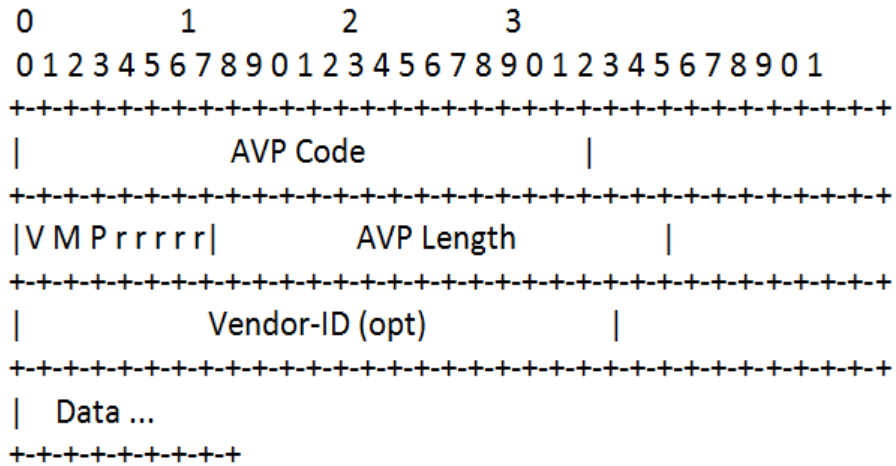


Figure 4.4 AVP header content [40]

The size of the AVP header is calculated as the sum of all the header's fields.

The details of the computations are found in Table 4.2.

Table 4.2 AVP header size

AVP Header Fields	Size (bits)
AVP Code	32
AVP Flags	8
AVP Length	24
Vendor-ID (opt)	32
Data	...
Total Size	96 + Data

Hence, the total size of the AVP header is $96 \text{ bits} + \text{Data}$, where the data size depends on the AVPs that belong to the diameter protocol. The information elements of these AVPs along with their type of encoding, description and size appear in Table 4.4. Note that if the AVP is of type grouped, the Data field is then specified as a sequence of AVPs and its total size is the sum of these AVPs.

Before discussing the table, it is good to mention here that for computation purposes, some assumptions are taken into consideration:

- host name is of 24 characters: *accesspoint7.example.com*
- realm name is of 11 characters: *example.com*

- the grouped Proxy-info AVP contains only 1 AVP of length 1 byte
- User-name is of type dot-string of an average of 4 characters according to rfc2486: *fred@example.com* [41], [42].
- The shape type chosen for “Location-Estimate” information element is *ellipsoid point*. The ellipsoid point is that of a point on the surface of the ellipsoid, and consists of a latitude and a longitude [43] as it is shown in Figure 4.5. This type can be well used in locating a mobile device in order to compare its proximity to another device.

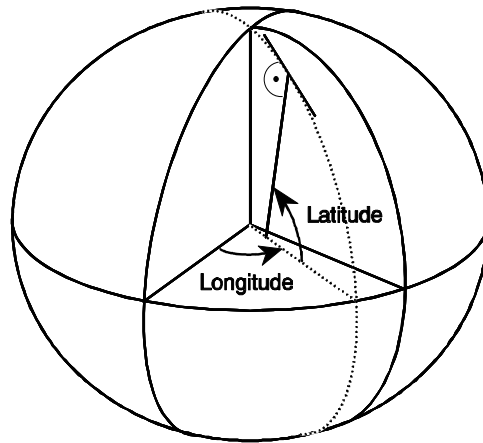


Figure 4.5 Description of a point as two coordinates [43]

The organization of “Location-Estimate” information element is depicted in Figure 4.6.

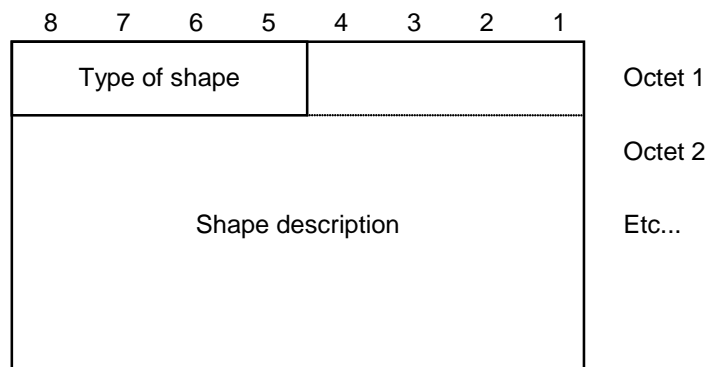


Figure 4.6 “Location-Estimate” information element content [43]

Table 4.3 represents the coding of the different type shapes. Since the type of shape chosen is “Ellipsoid point”, the corresponding coding will be “0000”.

Table 4.3 Coding of type shape [43]

Bits	
4 3 2 1	
0 0 0 0	Ellipsoid Point
0 0 0 1	Ellipsoid point with uncertainty Circle
0 0 1 1	Ellipsoid point with uncertainty Ellipse
0 1 0 1	Polygon
1 0 0 0	Ellipsoid point with altitude
1 0 0 1	Ellipsoid point with altitude and uncertainty Ellipsoid
1 0 1 0	Ellipsoid Arc
other values	reserved for future use

The coding of an ellipsoid point is described in Figure 4.7.

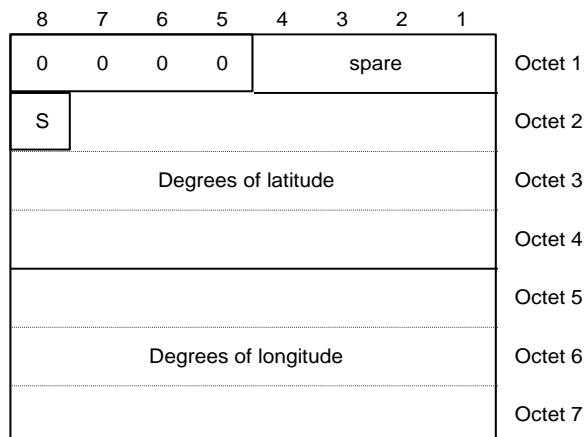


Figure 4.7 Shape description of a point [43]

The meaning of these fields can be summarized as follows: S, Sign of latitude (Bit value 0 for North indication while Bit value 1 for South), Degrees of latitude (Bit 1 of octet 4 is the low order bit), Degrees of longitude (Bit 1 of octet 7 is the low order bit). An example of “Location-Estimate: information element in Universal Geographical Area Description (GAD) shapes can be found in [43].

Now that we defined our assumptions, we can calculate the size of each information element as shown in Table 4.4.

Table 4.4 Diameter protocol information elements [40]

Information Element	Type of Encoding	Description	Size (bits)
Session-Id	UTF8String	<p>It is used to identify a specific session. Its recommended format:</p> <p><i><DiameterIdentity>;<high 32 bits>;<low 32 bits>[;<optional value>]</i></p> <p><i><high 32 bits></i> and <i><low 32 bits></i> are decimal representations of the high and low 32 bits of a monotonically increasing 64-bit value. DiameterIdentity is in ASCII form in order to be compatible with existing DNS infrastructure.</p> <p>Example, in which there is no optional value: accesspoint7.example.com;1876543210;523</p> <p>Example, in which there is an optional value: accesspoint7.example.com;1876543210;523;mobile @200.1.1.88</p>	$(24 \times 8) + 32 + 32 = 256$
Vendor-Specific-Application-Id	Grouped	<p>The Vendor-Id AVP is an informational AVP pertaining to the vendor who may have authorship of the vendor-specific Diameter application.</p> <p><i><Vendor-Specific-Application-Id> ::=</i> <i>< AVP Header: 260 ></i> <i>{ Vendor-Id } (Unsigned 32)</i> <i>[Auth-Application-Id] (Unsigned 32)</i> <i>[Acct-Application-Id] (Unsigned 32)</i></p>	$32 + 32 + 32 = 96$
Auth-Session-State	Enumerated Integer 32	<p>It specifies whether state is maintained for a particular session. <i>STATE_MAINTAINED 0 Or</i> <i>NO_STATE_MAINTAINED 1</i></p>	32
Origin-Host	DiameterIdentity	<p>It MUST be present in all Diameter messages. This AVP identifies the endpoint that originated the Diameter message. Its format is derived from the OctetString Basic AVP Format.</p> <p style="text-align: center;">DiameterIdentity = FQDN/Realm</p> <p><i>Fully Qualified Domain Name (FQDNs) are represented in ASCII form</i></p> <p>The realm is the string in the Network Access Identifier NAI that immediately follows the '@' character. It is used to determine whether messages can be satisfied locally or whether they must be routed or redirected.</p>	$24 \times 8 = 192$

Table 4.4 Diameter protocol information elements [40] (Continued)

Origin-Realm	DiameterIdentity	This AVP contains the Realm of the originator of any Diameter message and MUST be present in all messages	11×8 = 88
Destination-Host	DiameterIdentity	This AVP MUST be present in all unsolicited agent initiated messages, MAY be present in request messages, and MUST NOT be present in answer messages	24×8= 192
Destination-Realm	DiameterIdentity	It contains the realm to which the message is to be routed. The Destination-Realm AVP MUST NOT be present in answer messages	8×11= 88
Proxy-info	Grouped	It contains the identity and local state information of the Diameter node that creates and adds it to a message. A relay or proxy agent MAY include the Proxy-Info AVP in requests if it requires access to any local state information when the corresponding response is received. Proxy-Info ::= < AVP Header: 284 > { Proxy-Host } { Proxy-State } * [AVP] where Proxy-Host is of type DiameterIdentity. It contains the identity of the host that added the Proxy-Info AVP The Proxy-State is of type OctetString. It contains state information that would otherwise be stored at the Diameter entity that created it.	192+8+8= 208
Route-Record	DiameterIdentity	A relay or proxy agent MUST append a Route-Record AVP to all requests forwarded. A relay or proxy agent MUST check for forwarding loops when receiving requests. A loop is detected if the server finds its own identity in a Route-Record AVP.	24×8= 192
User-name	UTF8String		4×8= 32
Result-Code	Unsigned 32	All Diameter answer messages in IETF-defined Diameter application specifications MUST include one Result-Code AVP. The Result-Code data field contains an IANA-managed 32-bit address space representing errors.	32

Table 4.4 Diameter protocol information elements [40] (Continued)

Experimental-Result	Grouped	<p>It indicates whether a particular vendor-specific request was completed successfully or whether an error occurred.</p> <p>Experimental-Result ::= < AVP Header: 297 > { Vendor-Id } (Unsigned32) { Experimental-Result-Code } (Unsigned32)</p>	32+32= 64
Failed-AVP	Grouped	<p>It provides debugging information in cases where a request is rejected or not fully processed due to erroneous information in a specific AVP.</p> <p>A Diameter answer message SHOULD contain an instance of the Failed-AVP AVP that corresponds to the error indicated by the Result-Code AVP.</p> <p><Failed-AVP> ::= < AVP Header: 279 > 1* {AVP}</p>	8
Location-Estimate	OctetString	<p>It shall contain an estimate of the location of an MS in universal coordinates and the accuracy of the estimate. It is expressed in GAD shapes [43].</p> <p>A bit string encoding a geographical description shall consist of the following parts:</p> <p>Type of Shape: 4 bits + 4 bits spare Shape Description = 6×8 = 48 bits <i>(Assume ellipsoid point)</i></p>	56
Supported-Features	Grouped	<p>It may inform the destination host about the features that the origin host supports for the application [44].</p> <p>Supported-Features ::= < AVP header: 628 10415 > { Vendor-Id } (Unsigned32) { Feature-List-ID } (Unsigned32) { Feature-List } (Unsigned32) *[AVP]</p>	32+32+32= 96

Other information elements that belong to ProSe standards [45], [46] also appear in these diameter messages. The type of encoding used for these information elements as well as their corresponding description and size appear in Table 4.5. For simulation purposes, we assume that the EPUID requester consists of 4 characters.

Table 4.5 ProSe standards information elements

Information Element	Type of Encoding	Description	Size (bits)
Requesting-EPUID	UTF8string	It refers to an identifier for EPC-level ProSe Discovery	4×8= 32
Time window	Unsigned 32	It contains the maximum number of seconds of validity of the proximity request	32
App-Layer-User-Id	UTF8String	It contains an identity identifying a user within the context of a specific application (e.g. alice@social.net)	16×8= 128
Range	Enumerated Int 32		32
PRR-Flags	Unsigned 32	It contains a bit mask	32
PRA-Flags	Unsigned 32	It contains a bit mask	32
PLR-Flags	Unsigned 32	It contains a bit mask	32
PLA-Flags	Unsigned 32	It contains a bit mask	32
ProSe-Subscription-Data	Grouped	ProSe-Subscription-Data ::= <AVP header: xxx 10415> { ProSe-Permission-List } (<i>Unsigned 32</i>) *[PLMN-Allowed-Discovery] *[AVP] ProSe-Permission-List contains a bit mask set to 1 to indicate that the user is allowed to use EPC-level ProSe Discovery PLMN-Allowed-Discovery ::= <AVP header: xxx 10415> [Visited-PLMN-Id] (<i>OctetString</i>) [Discovery-Allowed] (<i>Unsigned32</i>) *[AVP] <i>Discovery-Allowed contains a bit mask that indicates if UE is authorized to announce or monitor or both</i>	32 + (8+32) = 72
Visited-PLMN-Id	OctetString	The ID of the visited Public Land Mobile Network [47]	8

Note that we created new AVPs in order to encapsulate the needed information for our system discovery. In creating these AVPs, we respected the standards where “a new AVP being defined MUST use one of the data types listed in the standard” [41]. For instance, for each message, we have included a Flag of encoding type: Unsigned 32, same type used in the standards. The “Provider-EPUID” as well as “Requesting-EPUID” are UTF-8 strings. The remaining information elements along with their type of encoding and sizes can be found in Table 4.6.

Table 4.6 Newly created AVPs

Information Element	Type of Encoding	Size (bits)
MRQ-Flags	Unsigned 32	32
PRQ-Flags	Unsigned 32	32
MRP-Flags	Unsigned 32	32
LRP-Flags	Unsigned 32	32
PRAK-Flags	Unsigned 32	32
DCM-Flags	Unsigned 32	32
Provider-EPUID	UTF8string	4×8 = 32
Provider-PFID	Unsigned 32	32
App ID	Unsigned 32	32
Keywords	UTF-8	5×(8×8) + 4×8= 352
Keywords-Indices	UTF-8	3×8 + 2×8 = 40

Some messages are found in the standards [45], [46]. Below is the format of each of these messages. Our newly added AVPs are marked in bold.

Proximity Request (2) [46]:

```
< ProSe-Proximity-Request > ::=
    < Diameter Header: CC5, REQ, PXY, 16777xxx >
    < Session-Id >
    [ Vendor-Specific-Application-Id ]
    { Auth-Session-State }
    { Origin-Host }
    { Origin-Realm }
    [ Destination-Host ]
    { Destination-Realm }
    *[ Supported-Features ]
    { PRR-Flags }
    { Requesting-EPUID }
    { Provider-EPUID }
    { Time-Window }
    { Location-Estimate }
    *[ AVP ]
    *[ Proxy-Info ]
    *[ Route-Record ]
```

ProSe-Subscriber-Information-Request (PIR) Command or Location Request [45]:

```
< ProSe-Subscriber-Information-Request > ::=
    < Diameter Header: xxx, REQ, PXY, xxxxxx >
    < Session-Id >
    [ Vendor-Specific-Application-Id ]
    { Auth-Session-State }
    { Origin-Host }
    { Origin-Realm }
    [ Destination-Host ]
    { Destination-Realm }
    { Provider-EPUID }
    { User-Name }
    *[ Supported-Features ]
    *[ AVP ]
    *[ Proxy-Info ]
    *[ Route-Record ]
```

ProSe-Subscriber-Information-Answer (PIA) Command or location response [45]:

```
< ProSe-Subscriber-Information-Answer > ::=
    < Diameter Header: xxx, PXY, xxxxxx >
    < Session-Id >
    [ Vendor-Specific-Application-Id ]
    [ Result-Code ]
    [ Experimental-Result ]
    { Auth-Session-State }
    { Origin-Host }
    { Origin-Realm }
    { Provider-EPUID }
    [ ProSe-Subscription-Data ]
    [ Visited-PLMN-Id ]
    *[ Supported-Features ]
    *[ AVP ]
    *[ Failed-AVP ]
    *[ Proxy-Info ]
    *[ Route-Record ]
```

ProSe Proximity Answer [46]:

```
< ProSe-Proximity-Answer > ::=
    < Diameter Header: CC5, PXY, 16777xxx >
    < Session-Id >
    [ Vendor-Specific-Application-Id ]
    [ Result-Code ]
    [ Experimental-Result ]
    { Auth-Session-State }
    { Origin-Host }
    { Origin-Realm }
    *[ Supported-Features ]
    { PRA-Flags }
    { Provider-EPUID }
    [ Location-Estimate ]
    *[ AVP ]
    *[ Failed-AVP ]
    *[ Proxy-Info ]
    *[ Route-Record ]
```


Note that [WLAN-Link-Layer-Id] was omitted since it is an AVP of conditional category and it is present only if the requesting UE has requested EPC support for WLAN direct discovery which is not the case in our scenarios.

ProSe-Location-Update-Request [46]:

```
< ProSe-Location-Update-Request > ::=
    < Diameter Header: CC6, REQ, PXY, 16777xxx >
    < Session-Id >
    [ Vendor-Specific-Application-Id ]
    { Auth-Session-State }
    { Origin-Host }
    { Origin-Realm }
    [ Destination-Host ]
    { Destination-Realm }
    *[ Supported-Features ]
    { PLR-Flags }
    { Provider-EPUID }
    { Location-Estimate }
    *[ AVP ]
    *[ Proxy-Info ]
    *[ Route-Record ]
```

Since not all the messages used in our system appear in the standards, we created new messages having a compatible format with the aforementioned ones.

In fact, there are two types of messages: message request and message response which can be defined explicitly in the Diameter header: “REQ” for a message request and a blank for the response. Another difference between these two types of messages is that the AVPs: “Origin-Host” and “Origin-Realm” appear in both types; however, only “Destination-Host” and “Destination-Realm” exist in the request messages. Therefore, to be compliant with the standards, we included the “REQ” in the header of the message requests and left it empty for the message responses. Moreover, we respected the AVP issue regarding the Hosts and Realms. Furthermore, we integrated the AVPs created in Table 4.6 to encapsulate the needed data. Note that we kept the AVPs related to the

Diameter protocol the same as in the standards. Going with this analogy, we represent below the format of the newly created messages.

Proximity Request (1):

```
< ProSe-Proximity-Request > ::=
    < Diameter Header: CC5, REQ, PXY, 16777xxx >
    < Session-Id >
    [ Vendor-Specific-Application-Id ]
    { Auth-Session-State }
    { Origin-Host }
    { Origin-Realm }
    [ Destination-Host ]
    { Destination-Realm }
    *[ Supported-Features ]
    { PRQ-Flags }
    { Requesting-EPUID }
    { Requesting-ALUID }
    { Application-Id }
    { Time-Window }
    { Location-Estimate }
    { Range }
    { Requesting-Keywords }
    *[ AVP ]
    *[ Proxy-Info ]
    *[ Route-Record ]
```

Map Request:

```
< ProSe-Map-Request > ::=
    < Diameter Header: CC5, REQ, PXY, 16777xxx >
    < Session-Id >
    [ Vendor-Specific-Application-Id ]
    { Auth-Session-State }
    { Origin-Host }
    { Origin-Realm }
    [ Destination-Host ]
    { Destination-Realm }
    *[ Supported-Features ]
    { MRQ-Flags }
    { Application-Id }
    { Requesting-Keywords }
    *[ AVP ]
    *[ Proxy-Info ]
    *[ Route-Record ]
```

Map Response:

```
< ProSe-Map-Response > ::=
    < Diameter Header: CC5, PXY, 16777xxx >
    < Session-Id >
    [ Vendor-Specific-Application-Id ]
    [ Result-Code ]
    [ Experimental-Result ]
    { Auth-Session-State }
    { Origin-Host }
    { Origin-Realm }
    *[ Supported-Features ]
    { MRP-Flags }
    *{ Provider-EPUID }
    *{ Provider-PFID }
    *{Keywords-Indices}
    *[ AVP ]
    *[ Failed-AVP ]
    *[ Proxy-Info ]
    *[ Route-Record ]
```

Proximity Request Ack (2):

```
< ProSe-Request-Ack > ::=
    < Diameter Header: CC5, PXY, 16777xxx >
    < Session-Id >
    [ Vendor-Specific-Application-Id ]
    [ Result-Code ]
    [ Experimental-Result ]
    { Auth-Session-State }
    { Origin-Host }
    { Origin-Realm }
    *[ Supported-Features ]
    { PRAK-Flags }
    *[ AVP ]
    *[ Failed-AVP ]
    *[ Proxy-Info ]
    *[ Route-Record ]
```

LCS Location Reporting Request:

```
< ProSe-LCS-Location Reporting-Request > ::=
    < Diameter Header: xxx, REQ, PXY, xxxxxx >
    < Session-Id >
    [ Vendor-Specific-Application-Id ]
    { Auth-Session-State }
    { Origin-Host }
    { Origin-Realm }
    [ Destination-Host ]
    { Destination-Realm }
    { Provider-EPUID } | | { Requesting-EPUID }
    *[ Supported-Features ]
    {LRQ-Flags}
    *[ AVP ]
    *[ Proxy-Info ]
    *[ Route-Record ]
```

LCS Location Report:

```
< ProSe-LCS-Location-Report > ::=
    < Diameter Header: xxx, PXY, xxxxxx >
    < Session-Id >
    [ Vendor-Specific-Application-Id ]
    [ Result-Code ]
    [ Experimental-Result ]
    { Auth-Session-State }
    { Origin-Host }
    { Origin-Realm }
    [ Location-Estimate ]
    *[ Supported-Features ]
    *[ AVP ]
    *[ Failed-AVP ]
    *[ Proxy-Info ]
    *[ Route-Record ]
```

Proximity Alert:

```
< ProSe-Proximity-Alert > ::=
    < Diameter Header: CC5, REQ, PXY, 16777xxx >
    < Session-Id >
    [ Vendor-Specific-Application-Id ]
    { Auth-Session-State }
    { Origin-Host }
    { Origin-Realm }
    [ Destination-Host ]
    { Destination-Realm }
    *[ Supported-Features ]
    { PRR-Flags }
    { Application-Id }
    *{ Provider-EPUID }
    *{Keywords-Indices}
    *[ AVP ]
    *[ Proxy-Info ]
    *[ Route-Record ]
```

Decision Message:

```
< ProSe-Decision-Message > ::=
    < Diameter Header: xxx, PXY, xxxxxx >
    < Session-Id >
    [ Vendor-Specific-Application-Id ]
    [ Result-Code ]
    [ Experimental-Result ]
    { Auth-Session-State }
    { DCM-Flags}
    { Origin-Host }
    { Origin-Realm }
    *[ Supported-Features ]
    { Application-Id }
    {Provider-EPUID}
    *[ AVP ]
    *[ Failed-AVP ]
    *[ Proxy-Info ]
    *[ Route-Record ]
```

After knowing the format of each message, we can now calculate the size of these packets by adding the size of each AVP and its corresponding header as well as the diameter header. The size of each packet in bits can be found in Table 4.7.

For computation purposes, some assumptions are made:

- To calculate the size, we only consider the mandatory AVPs. For instance, “supported features” is optional, hence it should not be counted in our computations. All flags are conditional but it shall be present only when the Result-Code AVP is DIAMETER_SUCCESS. We assume that there is always success, hence we should consider the flags in calculating the traffic.
- Failed-AVP is a conditional AVP containing the AVPs that caused the failure. Since we assume that all is success, we do not consider this information element in computing the traffic.
- Provider-EPUID consists of 4 characters.
- There are 5 keywords of 8 characters each, separated by a “;”.
- Assume there are 5 providers matching the keywords but only 3 are near.

Therefore, the list contains 3 providers. ($3 \times \text{size}(\text{Provider_EPUID})$ in the *Proximity Alert message*)

- Assume there are 3 matching keywords on average. By this, 3 indices separated by “;” are considered. ($3 \times \text{size}(\text{Keywords_Indices})$ in the *Proximity Alert message*)

Table 4.7 Packet's size

Packet	Size (bits)
Proximity Request (1)	3832
Map Request	3072
Map Response	2704
Proximity Request (2)	3032
Location Request	2624
Location Response	2584
Proximity Request Ack (1) or called ProSe Proximity Answer	2312
Proximity Request Ack (2)	2312
LCS Location Reporting Request	2624
LCS Location Report	2336
ProSe Location Update	2776
Proximity Alert	3416
Decision Message	2440

The next step for the traffic calculation is to find the number of times a message is sent between the entities. Some probabilities should be estimated for this purpose.

If L is the total number of PLMNs and l is the number of PLMNs that contain providers having matched keywords with the requesters, ω is the number of keywords matching applications at one provider, the probability of matching at least one keyword in order to include the provider in the list of candidate providers will be 0.63 according to the matching problem [48].

$$p(\text{matching keywords at 1 provider}) \cong 0.63$$

$$p_{\text{keywords}} = p(l \text{ out of } L \text{ PLMNs}) = \binom{L}{l} (0.63)^l (0.37)^{L-l}$$

To get the 0.63, we assume that the number of keywords saved in the application server database corresponding to a provider is equal to the number of keywords submitted by the requester.

The probability of having the PLMN of the provider different from the one of the requester can be written as follows:

$$p_{interPLMN} = p(ProSe\ i \rightarrow ProSe\ j, i \neq j) = 1 - p_{intraPLMN}$$

$$p_{intraPLMN} = \frac{l}{2^l}$$

In order to calculate the traffic flowing between the ProSe and its corresponding SLP, we start by defining some parameters. Note that many providers can belong to the same ProSe as it is shown in the Figure 4.8.

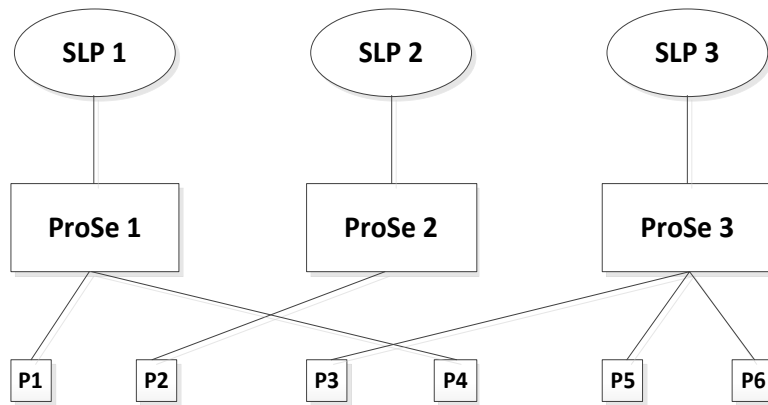


Figure 4.8 Providers distribution

Let U be the number of UEs, v the average speed, T_m the time to move $D_{max}/2$ meters, T_r request period, p_{picked} the probability that a ProSe Function is picked by a UE, N_r number of request in T_m per UE, p_m the probability that 2 or more UEs requesting in T_m location updates about same provider.

$$T_m = \left(\frac{D_{max}}{2}\right)/v \text{ seconds}$$

$$T_r = \frac{1}{r}$$

$$N_r = \frac{T_m}{T_r}$$

$$p_{picked} = \frac{N_r + 1}{2 \times l}$$

$$p_m = 1 - p \text{ (0 or 1 UE requesting same provider in } T_m)$$

$$p_0 = \binom{N}{0} (p_{picked})^0 (1 - p_{picked})^{N-0}$$

$$p_1 = \binom{N}{1} (p_{picked})^1 (1 - p_{picked})^{N-1}$$

$$p_m = 1 - p_0 - p_1$$

Therefore, the traffic between the SLP and ProSe will be multiplied by 1 minus the probability of having 2 or more UEs requesting location updates about the same provider. Some parameters used in the equations are defined in Table 4.8.

Table 4.8 Parameters used in the equations

Parameter	Definition
N	Number of requesters
r	Request rate
T_r	Request period
L	Number of PLMNs
N_r	Number of request in T_m per UE
M	Number of all providers
K	Number of providers holding a particular application
U	Number of UEs
l	Number of ProSe Matching
v	average speed
T_m	Time to move $D_{max}/2$ meters

In the following tables, we estimate the number of times a message is sent between the entities along with their corresponding size already calculated in Table 4.7. We divide the messages into two parts: inner traffic (Table 4.9) which is related to

messages exchanged between the core entities and wireless medium traffic (Table 4.10) which corresponds to the messages exchanged between the UE and the core network. Note that the sequence diagram showing all these LTE messages used to transport the request and reply information between the different core entities was previously presented in Figure 3.15.

Table 4.9 Packets' size and number of times they are sent (inner traffic)

Packet	Number of times sent	Size (bits)
Map Request	$N \times r$	3072
Map Response	$N \times r$	2704
Proximity Request (2)	$N \times r \times p_{Keywords} \times p_{interPLMN}$	3032
Location Request	$N \times r \times p_{Keywords}$	2624
Location Response	$N \times r \times p_{Keywords}$	2584
Proximity Request Ack (1) or called ProSe Proximity Answer	$N \times r \times p_{Keywords} \times p_{interPLMN}$	2312
Proximity Request Ack (2)	$N \times r$	2312
LCS Location Reporting Request	$(1 - p_m) \times N \times r \times p_{Keywords}$	2624
LCS Location Report	$(1 - p_m) \times N \times r \times p_{Keywords}$	2336
ProSe Location Update	$N \times r \times p_{Keywords} \times p_{interPLMN}$	2776

Table 4.10 Packets' size and number of times they are sent (wireless medium traffic)

Packet	Number of times sent	Size (bits)
Proximity Request (1)	$N \times r$	3832
Proximity Alert	Case 1: $N \times r \times p_{Keywords}$ Case 2: $N \times r \times (1 - p_{Keywords})$ Case 1: Code Ok List: : App ID EPUID_Providers Keyword_Provider Case 2: Notification Code	3416
Decision Message	$N \times r \times p_{Keywords}$	2440

To study the parameters affecting the two types of traffic in the network, i.e. inner traffic and wireless medium traffic, we conducted many experiments. Our default network, during these experiments, contains a total of 4 PLMNs where only 2 represent matching ProSe Functions. This means that the providers having the matching keywords with the requester's keywords belong to these 2 PLMNs. The requester is a pedestrian moving with an average velocity of 1.4 m/s and sending one request per second. We consider the maximum distance separating two UEs defined to be in proximity equal to 25 m [23].

In the first experiment, we change the number of requesters between 2 and 20 in a step of 2 and keep the other parameters according to the default values. We plot the traffic versus the number of requesters in Figure 4.9.

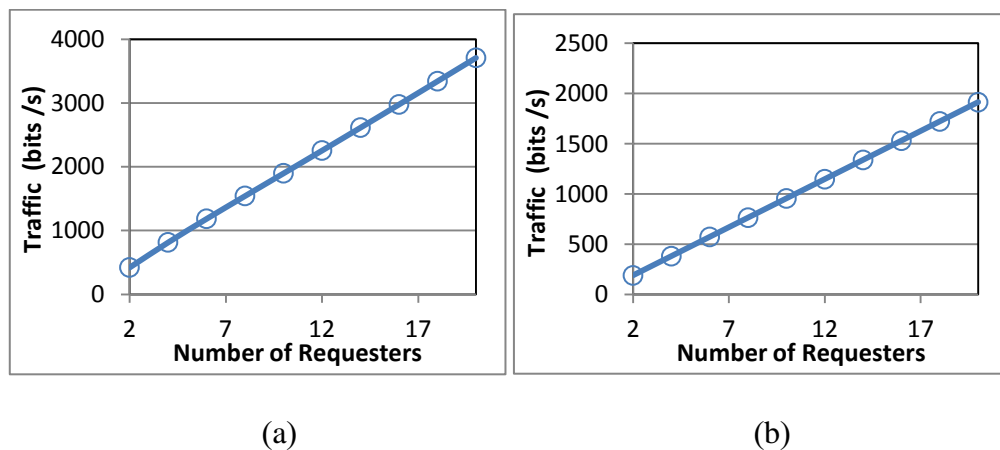


Figure 4.9 Number of requesters' effect on the traffic (a) inner traffic (b) wireless medium traffic

As the graphs show, the traffic increases with the increase of the number of requesters. It starts by around 500 bits/s (in case of inner traffic) and 200 bits/s (in case of wireless medium traffic) for 2 requesters then it reaches 3900 bits/s (in case of inner traffic) and 2000 bits/s (in case of wireless medium traffic) for 20 requesters. This can be analyzed by the fact that the more requesters exist in the network, the more the

number of packets (requests) sent in the wireless medium (i.e, wireless medium traffic) are to be processed by the network (i.e, inner traffic) and hence the traffic boosts.

In the next experiment, we fix the number of requesters participating in the network to 10 and change the number of requests sent per second per UE between 1 and 5. Hence, r will be equal be between $\left(\left(\frac{1}{60}\right) \times 1\right) = 0.0167$ and $\left(\left(\frac{1}{60}\right) \times 5\right) = 0.083$.

The other parameters' values remain the same. In Figure 4.10, we represent the traffic variation versus the request rate.

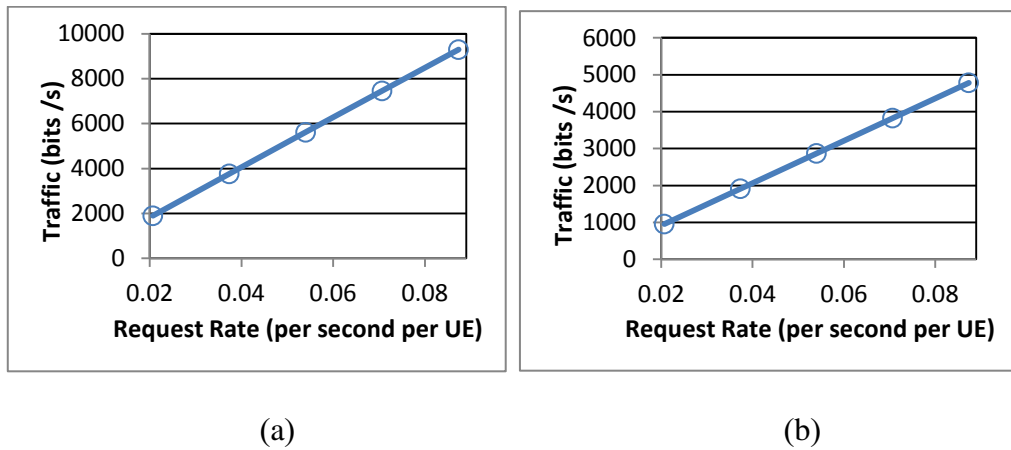


Figure 4.10 Request's rate effect on traffic (a) inner traffic (b) wireless medium traffic

As the graphs show, the traffic increases with the increase of the request rate. For 1 request per second per UE, the traffic is 2000 bits/s (in case of inner traffic) and 1000 bits/s (in case of wireless medium traffic) while it reaches 9800 bits/s for 5 requests per second per UE (in case of inner traffic) and 5000 bits/s (in case of wireless medium traffic). It is evident that when the number of requests sent by the request increases, the traffic in the network inflates.

Now, we fix again the request rate to 1 request per second per UE and change the number of matching prose functions between 1 and 3 as shown in Figure 4.11. In

this graph, we can see that the traffic increases with the augmentation of the number of matching ProSe Functions. It starts by around 1520 bits/s for only 1 matching ProSe Function and grows to reach around 2100 bits/s for 3 matching ProSe Functions in case of inner traffic while it starts by 760 bits/s for 1 matching ProSe Function in case of wireless medium traffic and reaches 1000 bits/s for 3 matching ProSe Functions. This can be analyzed by the fact that when the providers having matching keywords belong to different PLMNs, their corresponding ProSe Functions will be asked by the requester's ProSe Function in order to check their current positions. However, if these providers are registered in the same ProSe Function, the requester's ProSe Function will contact only this entity (one ProSe Function) and hence the traffic is lower.

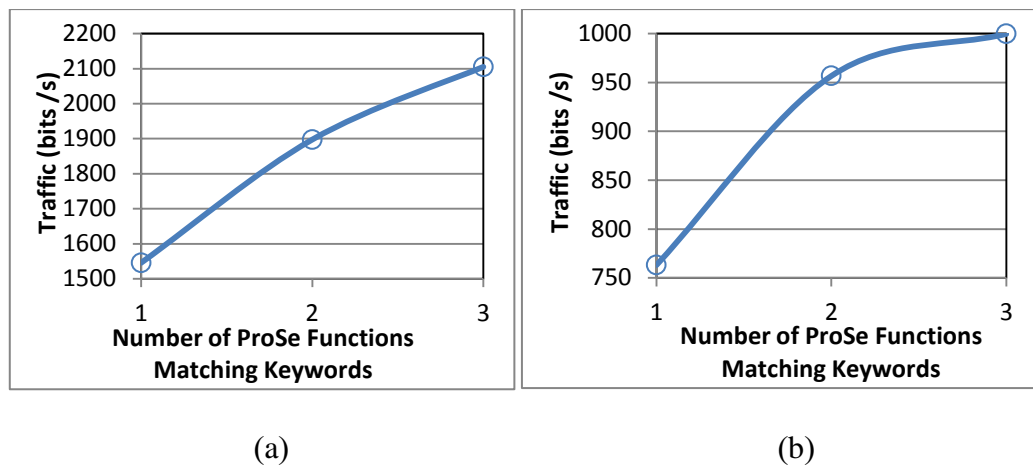
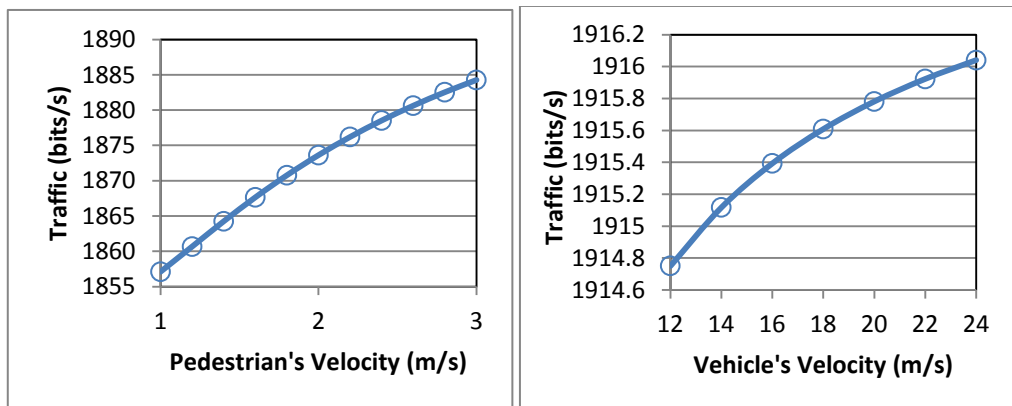


Figure 4.11 Number of Matching ProSe Functions's effect on the traffic (a) inner traffic (b) wireless medium traffic

In the last experiment, we fix back the number of matching ProSe Functions to 2 and change the requesters' speed. We consider here two scenarios: in the first scenario, the requesters are pedestrians while in the second one they are driving vehicles. For the first case, we vary the speed between 1 and 3 m/s in Figure 4.12 (a) while we change it between 12 and 24 m/s for the second case, in Figure 4.12 (b). In both cases, the traffic grows exponentially with the increase of the requesters' velocities. On a hand, in the

pedestrians' scenario, the traffic starts by 1857 bits/s for a requester walking at 1 m/s and reaches a value of 1885 bits/s for a requester walking at 3m/s. On the other hand, in the scenario where the requesters are driving vehicles, the traffic expands from 1914.6 bits/s for a requester driving at a normal speed equal to 12 m/s to 1916 bits/s for a requester moving at much higher speed. This can be analyzed by the fact that when the requester moves fast, he needs to update the network about his current location more frequently increasing by this the traffic in the network. It is good to mention here that we have only considered the inner traffic and not the wireless medium traffic since $p_{Keywords}$ does not depend on the velocity. Moreover, we have excluded the "location update" message from our computations because these will be required by the LTE system overall and there will be other applications using it, besides our system [35].



(a)

(b)

Figure 4.12 (a) Traffic for a pedestrian requester (b) Traffic for a requester driving a vehicle.

2. *Discovery Effectiveness*

In this section, we study the sources of errors that could occur at the search engine side (added to the Application Server) due to many causes:

- Number of keywords entered by the requester

The keywords used by the requester depend on his knowledge about the service. He may enter vague or targeted keywords that would affect on the result of the candidate providers sent back by the network (could contain irrelevant results relatively to his intended one). That is because several services belonging to the same category may share the same keywords. For instance, Service_1 and Service_2 belong to “Restaurants” category but differ by the type of cuisine they offer (Service_1: “Fast Food”, Service_2: “Vegetarian Food”). These services may share common keywords like “Restaurant”, “Food”... Therefore, when a requester enters “Restaurant” as a keyword to search for a “Fast Food” restaurant, he will get Service_2 (“Vegetarian Food”) as a provider candidate though this service was not in his intention. To represent this probability of error in our analysis, the keywords of the services having similar ones are concatenated into lists called “Concatenated lists”. The similarity between these two services (in terms of number of common keywords) is compared to a threshold. If the number of these common keywords is above this threshold, the services’ keywords will be concatenated, otherwise no aggregation will occur. Further details about the requester’s intention will be discussed in the next chapter.

- Number of keywords registered by the providers in the network to tag their services

When the provider registers his service in the network, he chooses a number of keywords that help the requester in finding him. However, there is a probability that the

requester uses keywords other than the ones selected by the providers to tag their services, and hence these providers will not be discovered.

- Number of providers in the network

Many providers can be offering different services in the network. However, there is a probability that these services are not the ones intended by the requesters.

Besides all these sources of errors, the requesters' speed and the size of the area they are moving in, influence also the probability of finding a provider in the requesters' proximity. In our analysis, we consider the providers fixed at predefined positions while the requesters moving according to Random Waypoint Mobility model (RWP) [49]. We took this assumption since the providers should be more or less stable in order to stay connected with the requester and offer their services. The RWP is a very popular and commonly used mobility model describing the movement behavior of a mobile node in a given system area: a node chooses randomly a destination point in this area and moves with a constant speed to reach it, then it pauses there for a certain time, chooses another destination point and speed and so on.

For proper nomenclature, several variables must be defined. The sets are written in upper case letters whereas the corresponding samples are written in lower case. In order to make it clearer, we will divide the analysis process into steps.

Step 1: Concatenated list corresponding to the requester's intention

To find the concatenated list containing the service intended by the requester, we compare each of the requester's keywords to all the constructed concatenated lists using the ranking function. This formula is widely adopted in the literature to measure the relevance scores of matching files to a given query in information retrieval [51]:

$$score(y_q, cl_m) = \frac{1}{|cl_m|} \times \left(1 + \ln f_{y_q}^{cl_m}\right) \times \left(1 + \frac{M}{f_{y_q}}\right)$$

where y_q is one of the Q requester's keywords ($Y = \{y_1, y_2, \dots, y_Q\}$), cl_m denotes one of the M concatenated lists ($Cl = \{cl_1, cl_2, \dots, cl_M\}$) while $|cl_m|$ is the length of cl_m , obtained by counting the number of keywords in this list. $f_{y_q}^{cl_m}$ indicates the term frequency of keyword y_q in cl_m , f_{y_q} designates the number of concatenated lists that have y_q in it.

However, the equation aforementioned calculates the score per keyword only. Therefore, the score of each cl_m over the Q keywords in the request will be

$$score(cl_m) = \sum_{q=1}^Q score(y_q, cl_m)$$

Only the one having the maximum score will be the intended concatenated list (Ct):

$$Ct = \max_m (score(cl_m))$$

Step 2: Services having matching keywords with the request

After finding the intended concatenated list (Ct), we compute the relevance score of each keyword along the C concatenated services (ct_c) in (Ct) ($Ct = \{ct_1, ct_2, \dots, ct_C\}$) using the equation below:

$$score(y_q, ct_c) = \frac{1}{|ct_c|} \times (1 + \ln f_{y_q}^{ct_c}) \times \ln(1 + \frac{C}{f_{y_q}})$$

where y_q denotes each keyword present in the request sent by the requester, ct_c is one of the total number of C services in Ct and $|ct_c|$ is the number of keywords in ct_c . It is good to mention here that the number of services in the concatenated list depends on a threshold, which is the minimum required number of common words to consider two services as similar (Further details will be discussed in the next chapter).

$f_{y_q}^{ct_c}$ is the term frequency of keyword y_q in ct_c (can be 1 or 0 whether the keyword exists or not in this service's pool) while f_{y_q} denotes the number of services that contain keyword y_q .

Having Q keywords y_q , the score of each service will be equal to:

$$score(ct_c) = \sum_Q score(y_q, ct_c)$$

The result will be a list of R matching services SR :

$$SR = \{sr_1, sr_2, \dots, sr_R\} = \{sr_r | (score(ct_c)) \geq 1\}$$

In order to study the effect of the number of concatenated services in a list on the score of each keyword per service, we assume that the average number of keywords is 22 per service ($|ct_c| = 22$), the keyword y_q exists in ct_t ($f_{y_q}^{ct_c} = 1$) and only one service contains it ($f_{y_q} = 1$). We plug these values in the equation of $score(y_q, ct_c)$ and plot the score of each keyword per service versus the number of concatenated services in Figure 4.13.

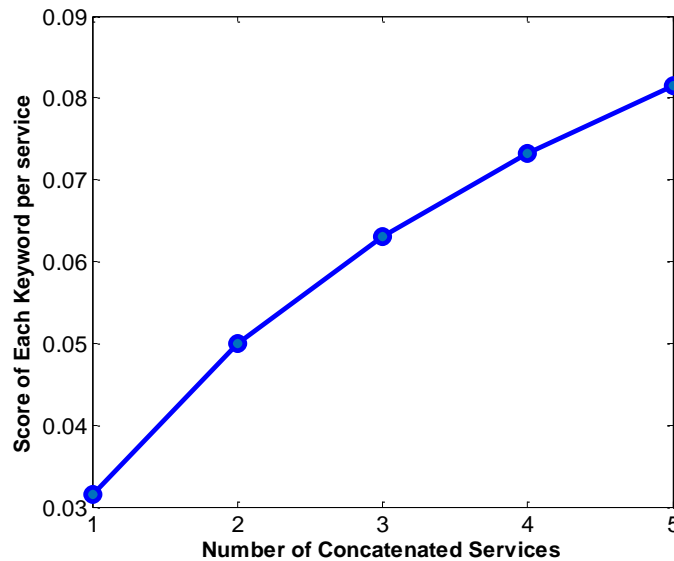


Figure 4.13 Score of each keyword per service versus the number of concatenated services

As the plot illustrates, the score of each keyword per service increases with the increase of the number of concatenated services. The likelihood of matching user keywords with services increases as the number of services per service category increases. This is justifiable since there are more similar services in this category in which the user is interested, and hence, his or her keywords are more likely to match.

Step 3: Providers choosing their services

Now that the requester's services are known, we need to match them with the providers' services. Each provider pv_p from P providers ($PV = \{pv_1, pv_2, \dots, pv_P\}$) chooses randomly a service sp from 50 services, where their choice depends on the popularity of this service. This can be illustrated by choosing from a cumulative pool depending on the occurrences of the service as shown in Figure 4.14.

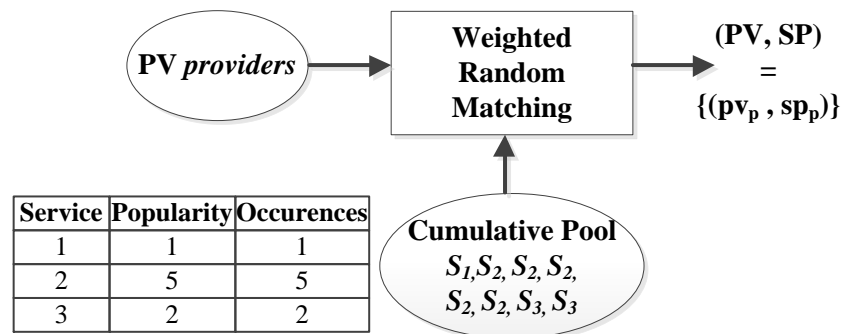


Figure 4.14 Services' popularities and providers' choices

For instance, service_2 which has a popularity of 5 will appear 5 times in the cumulative pool (S_2, S_2, S_2, S_2, S_2) as depicted in the figure above. The result will be a list of providers pv_p with their corresponding services sp_p :

$$(PV, SP) = \{(pv_1, sp_1), (pv_2, sp_2), \dots, (pv_P, sp_P)\}$$

Step 4: Providers choosing their keywords

Each provider pv_p who has already chosen his service sp_p in Step 4, chooses randomly W keywords K from sp_p 's keywords set K^{sp_p} .

$$pv_p \rightarrow (K)_p = (\{k_1, k_2, \dots, k_W\})_p \mid k_w = K_w^{sp_p}$$

By this, the list of providers contains the provider's ID pv_p , the service sp_p he has chosen and the keywords $(K)_p$ he has selected:

$$(PV, SP, (K)) = \{(pv_1, sp_1, (K)_1), (pv_2, sp_2, (K)_2), \dots, (pv_p, sp_p, (K)_p)\}$$

Step 5: Providers having matching services

Now, we need to compare the requester's services to the providers' chosen services in order to check which provider is offering the intended ones. In other terms, we need to compare $SR = \{sr_r\}$ to $SP = \{sp_p\}$ as illustrated in Figure 4.15.

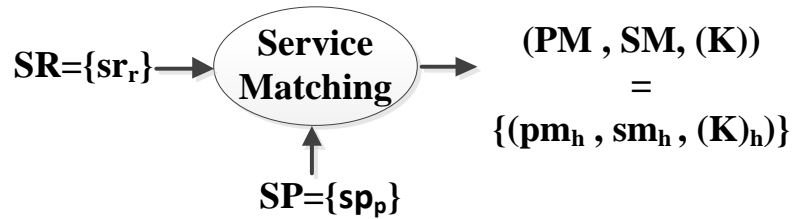


Figure 4.15 Service matching between requester and provider

The providers PM matching the requested services will be the set of H providers ($PM = \{pm_1, pm_2, \dots, pm_H\}$) where $\{pm_h\} = \{\forall pv_p \mid \{sp_p\} \in \{sr_r\}\}$ with their corresponding services SM and keywords

$$(PM, SM, (K)) = \{(pm_1, sm_1, (K)_1), (pm_2, sm_2, (K)_2), \dots, (pm_H, sm_H, (K)_H)\}$$

where $(K)_h = (\{k_1, k_2, \dots, k_W\})_h$. Note here that the number of providers having matching services depends on the number of requester's keywords. That's can be explained by the fact that when the requester enters more keywords, the number of matching services $\{sr_r\}$ increases and hence a larger variety of services will be searched

for among the providers' services which may lead to higher probability of matching. Moreover, the number of H providers also depends on the total number of providers in the network. That's because the greater the number of providers in the network, the larger the probability of finding the intended services.

Step 6: Providers having matching services and matching keywords

Now that the providers offering the services are known along with the keywords they have selected from this service's pool, we need to compare these keywords

$(K)_h = \{(k_w)_h\}$ against the requester's ones $Y = \{y_q\}$ as illustrated in Figure 4.16.

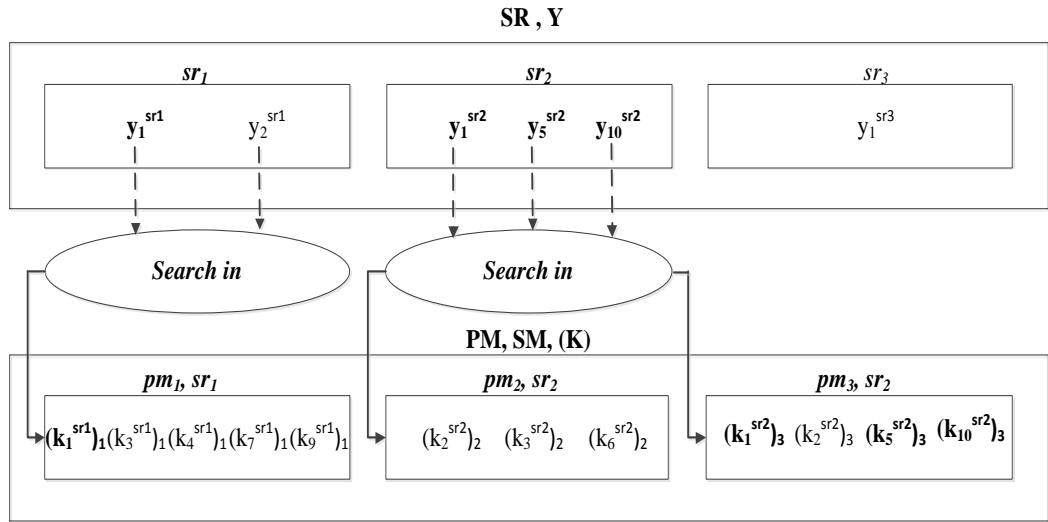


Figure 4.16 Keywords matching between requester and provider

For instance, we compare the keywords of the 3 services sr_1, sr_2, sr_3 present in the request to the matching providers (output of Step 5). Assume 3 matching providers pm_1, pm_2, pm_3 having sr_1, sr_2, sr_2 as matching services respectively. Since the services are matching ones, we mean by y same as k and $sm_1 = sr_1, sm_2 = sr_2, sm_3 = sr_2$. We can see that sr_2 has 2 providers offering it while sr_3 has none. The requester chose the first and second keywords of service sr_1 denoted as $y_1^{sr_1}, y_2^{sr_1}, y_5^{sr_1}$ while the provider pm_1 selected the first, third, fourth, seventh and ninth keywords of sr_1 . Therefore, while

searching for the requester's keywords in the providers' keywords, only $k_1^{sr_1}$ or (marked in bold) will be found. Note that the requester and the provider may choose the same service but selecting different keywords and hence this provider will not be considered as a matching provider (case of the provider pm_2). Note here that Step 5 helped in minimizing the processing time and complexity of the searching process. That's because the network is searching now only among the providers having the matching services and not among the total number of providers (filter a subset out of a set).

The matching weight of each provider (number of keywords matched) can be calculated using the ranking function [51]:

$$score((k_w)_h, sr_r) = \frac{1}{|sr_r|} \times (1 + \ln f_{(kw)_h}^{sr_r}) \times \ln(1 + \frac{R}{f_{(kw)_h}})$$

where $(k_w)_h$ stands for the keyword w belonging to provider h , sr_r is a service r in the set of R intended services while $|sr_r|$ is its length, meaning the number of keywords in it. $f_{(kw)_h}^{sr_r}$ designates the term frequency of the keyword $(k_w)_h$ in sr_r while $f_{(kw)_h}$ indicates the number of services that contain keyword $(k_w)_h$ in it. By this, we get the score of each keyword in the provider's pool. In order to calculate the provider's score, we sum the scores of all his keywords and hence it depends here on the number of keywords chosen by the provider: the more the number of keywords, the greater the score.

$$score(pm_h) = \sum_{w=1}^W score((k_w)_h, sr_r)$$

The number of matching providers will be equal to

$$|pm_h|/score(pm_h) > 1$$

Note here that not all the providers pm_h resulting from step 5 appear in this list again because the provider may have the service but did not use the adequate keywords while tagging himself as a provider for this service; and hence he will not be found since his $score(pm_h)$ is equal to 0.

In order to study the effect of the number of requester's keywords on the score, we assume that each two keywords in the request correspond to a certain service ($|sr_r| = 2$), only one service contains these keywords ($f_{(kw)_h} = 1$) and appears only once in the service's pool ($f_{(kw)_h}^{sr_r} = 1$). We vary the number of intended services (which increases with the number of requester's keywords) and plot its effect on the score in Figure 4.17.

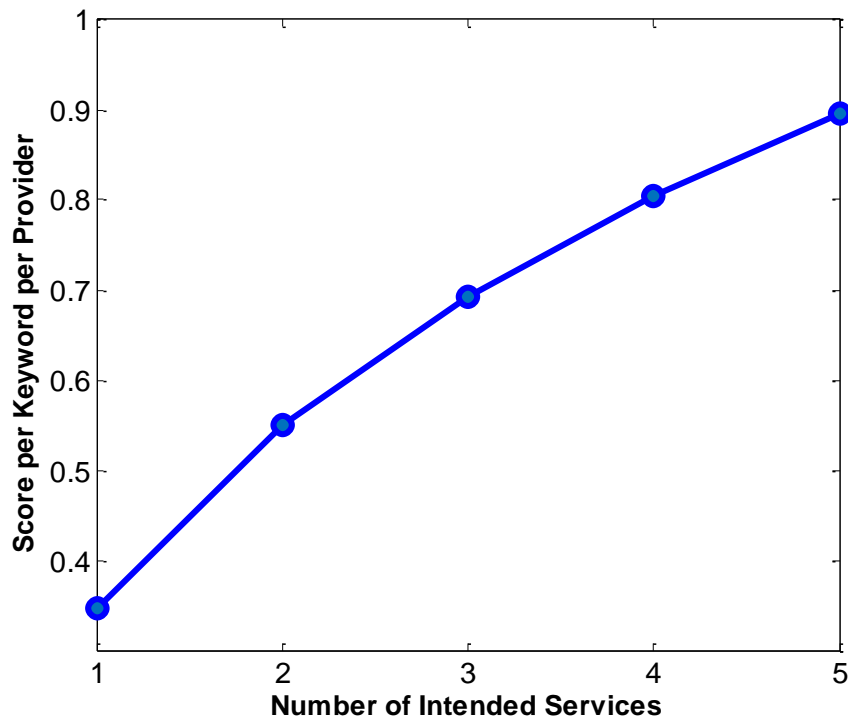


Figure 4.17 Score per keyword per provider versus the number of intended services

Step 7: Providers having matching services and matching keywords and being in proximity

In this step, the requester's position should be compared to the position of each provider pm_h found in Step 6. We use the analytical expression in [50] for the spatial node distribution $f_{XY}(x, y)$ of Random Waypoint movement process to approximate the distribution in a square area of size $a \times a$.

As it was proved in [50], this distribution is independent of the speed of nodes.

$$f_{XY}(x, y) \approx \frac{36}{a^6} \left(x^2 - \frac{a^2}{4}\right) \left(y^2 - \frac{a^2}{4}\right)$$

In Figure 4.18, we illustrate how a requester R can reach the provider's pm_h proximity in a simulation area of dimensions $a \times a$. The provider's proximity is represented by a circle around pm_h of radius D_{max} . As we can see, the requester can change directions, pause for a certain time, and then continue moving to reach pm_h .

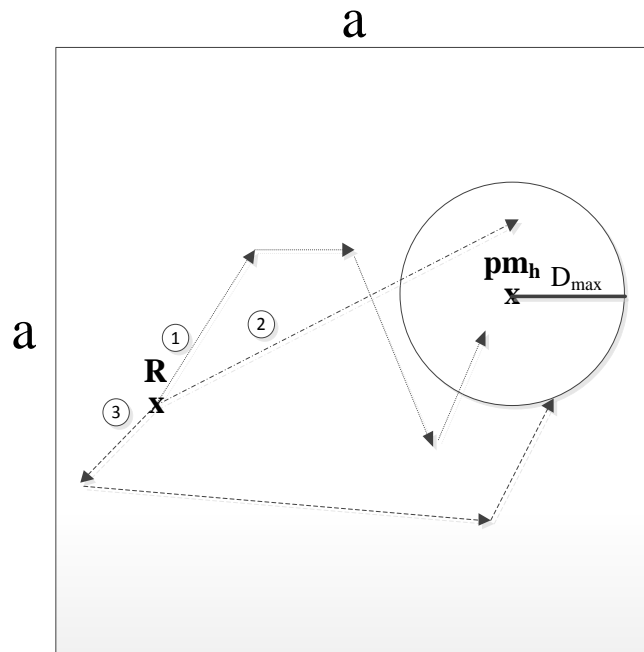


Figure 4.18 Requester changing directions to reach provider pm_h

Therefore, the probability that a requesting node is located in a certain subarea $A' \in A$ can be computed by integrating $f_{XY}(x, y)$ over this subarea where A' is the proximity of provider pm_h . The boundaries of this subarea can be represented as follows:

$$\begin{aligned} -D_{max} &\leq x \leq D_{max} \\ -\sqrt{D_{max}^2 - x^2} &\leq y \leq \sqrt{D_{max}^2 - x^2} \end{aligned}$$

Hence, the probability that a requester exists in A' can be written as:

$$P(\text{node in } A') = \int_{-D_{max}}^{D_{max}} \int_{-\sqrt{D_{max}^2 - x^2}}^{\sqrt{D_{max}^2 - x^2}} \frac{36}{a^6} \left(x^2 - \frac{a^2}{4}\right) \left(y^2 - \frac{a^2}{4}\right) dy dx$$

Solving this double integral leads to the following result:

$$P(\text{node in } A') = \frac{3\pi}{2} \left(\frac{D_{max}}{a}\right)^6 - \frac{9\pi}{2} \left(\frac{D_{max}}{a}\right)^4 + \frac{9\pi}{4} \left(\frac{D_{max}}{a}\right)^2$$

Note that this probability corresponds to a requester appearing in the proximity of only one provider. Therefore, $P(\text{node in } A')$ should be multiplied by the total number of providers pm_h resulting from Step 6 in order to find the probability of finding R in the proximity of all the providers having matching keywords with the request.

Figure 4.19 shows that the number of providers discovered in proximity decreases with the increase of the length of the area. If the requester sends a request when it is still far away from the provider (300m), the probability of finding him is too small and thus the number of providers found is approximately 0. However, when the requester approaches more and more (50m) from the region where the provider exists, the probability of discovering providers having matching keywords increases and thus the number of providers found becomes greater (3).

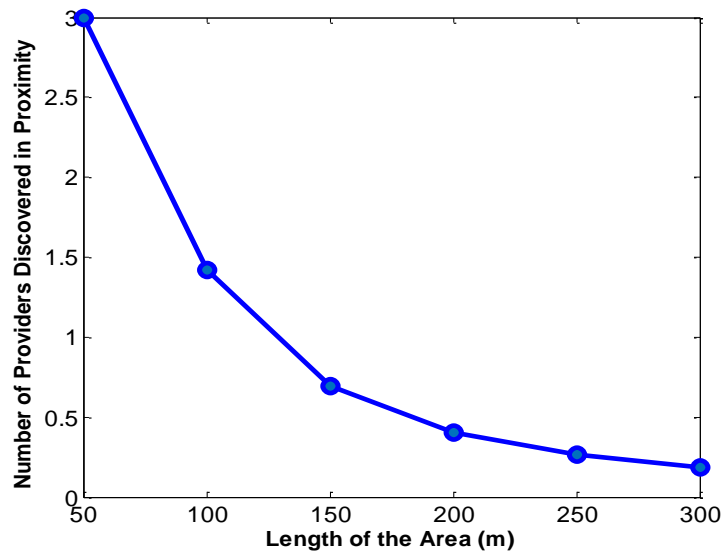


Figure 4.19 Number of providers discovered in proximity versus the length of the area

The entire system design can be summarized in the sequence diagram illustrated below:

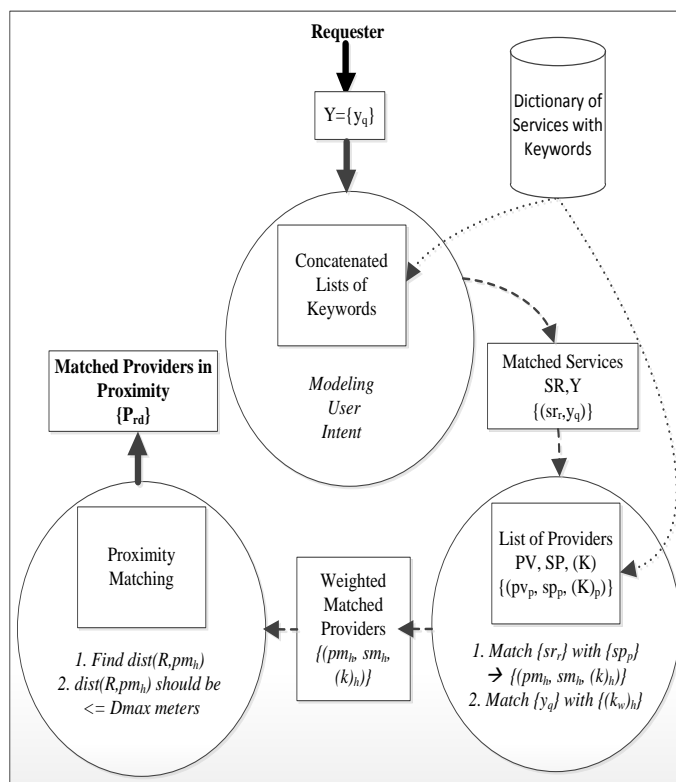


Figure 4.20 Sequence diagram of the entire system design

CHAPTER V

EXPERIMENTAL RESULTS

A. Requester's Intention

To study the performance of our system, we simulated the components and functions of the framework in Matlab, and created a pool of 50 services with associated (and in some cases overlapping) keywords (plus their synonyms). When registering a service, a provider randomly chooses one of the 50 services, and then a random number of keywords from the chosen service's keywords. When a requester looks for a particular service, the simulated network checks for similarity against services in the pool using the Minimum Edit Distance algorithm. Upon the start of the simulation, the requesters start moving within a $400 \times 400 \text{ m}^2$ area according to the Random WayPoint mobility pattern while the providers stay fixed at predefined positions.

Because of overlapping keywords among services, and among service providers, a request may result at times in false positives whose number also depends on the requester's simulated uncertainty and experience in choosing the right keywords. The 50 services are defined by a set of keywords gathered from Sensor Tower [52], and expanded by the keywords' synonyms obtained using Microsoft's Office automation tools [53]. These services categories were selected depending on their popularity. For instance, we consider 7 services for "Social Media" category knowing that it is one of the main applications for D2D technology along with file sharing and gaming. It is good to mention here that the keywords can be of any type: verb, noun or adjective, simulating by this the user's freedom in choosing his or her words.

Table 5.1 provides some statistics about the generated services, including the numbers of keywords and synonyms per service. It is good to mention here that the difference in the number of words denoted per service for each category in the 3rd column of this table is due to the dependency on the number of synonyms retrieved from the server. The average number of keywords is found to be 22 while the maximum and minimum are 41 and 11 respectively. After this step, repeating words were removed (repetition resulting from [53] for keywords that are also synonyms), so that each service is described by a set of distinct words. For clarification, we mean by using the term “keywords” in the rest of the thesis, the “keyword” itself or its “synonym”.

Table 5.1 Services and associated information

Category	Number of Services	Number of Keywords per Service
Restaurants	10	[22,14,17,28,12,41,18,21,22,20]
Music	7	[17,16,19,21,14,23,20]
Social Media	7	[25,19,22,21,31,20,33]
Real-Time News	4	[16,18,37,25]
School Tutoring	5	[29,19,21,34,21]
Games	4	[28,18,24,11]
Weather Services	2	[20,15]
Language Translation	3	[17,29,21]
Parking	4	[18,25,31,23]
Traffic	4	[40,23,20,18]
Total	50	

It was natural to end up with keywords that are shared by several services. For this reason, we measured the similarity among the services using Minimum Edit Distance algorithm or so called Levenshtein Distance (LD). This algorithm describes how similar or dissimilar two strings are by calculating the minimum number of “character edit operations” needed to turn one string into the other. These operations can be to delete, insert or substitute a character into this string.

The pseudo code for the Minimum Edit Distance algorithm can be found below [54]:

```

int LD(String s, String t) {
  m = s.length
  n = t.length
  int[m][n]
  d = empty int table for i from 0 to m
  d[i, 0] := i for j from 0 to n
  d[0, j] := j
  for j from 1 to n
    for i from 1 to m
      if s[i] = t[j] then d[i, j] := d[i-1, j-1] else d[i, j] := minimum(d[i-1, j] + 1, d[i, j-1] + 1, d[i-1, j-1] + 1)
  return d[m, n]}

```

Applying this algorithm, we generated a 50×50 similarity matrix as illustrated in Figure 5.1 where s_k and s_l are two different set of services' keywords and $g_{k,l}$ is the weight of similarity between them.

		S_l		
S_k			g_{k,l}	

Figure 5.1 Similarity matrix

This matrix is a symmetric matrix of 50×50 dimensions due to the fact that comparing service k to service l is the same as comparing service l to service k . We mean by the weight of similarity, the normalized similarity number which is divided it by the length of the union of service k and service l that can be found in the equation below:

$$length(s_k \cup s_l) = length(s_k) + length(s_l) - g_{k,l}$$

The pseudo code of the proposed algorithm can be written as follows:

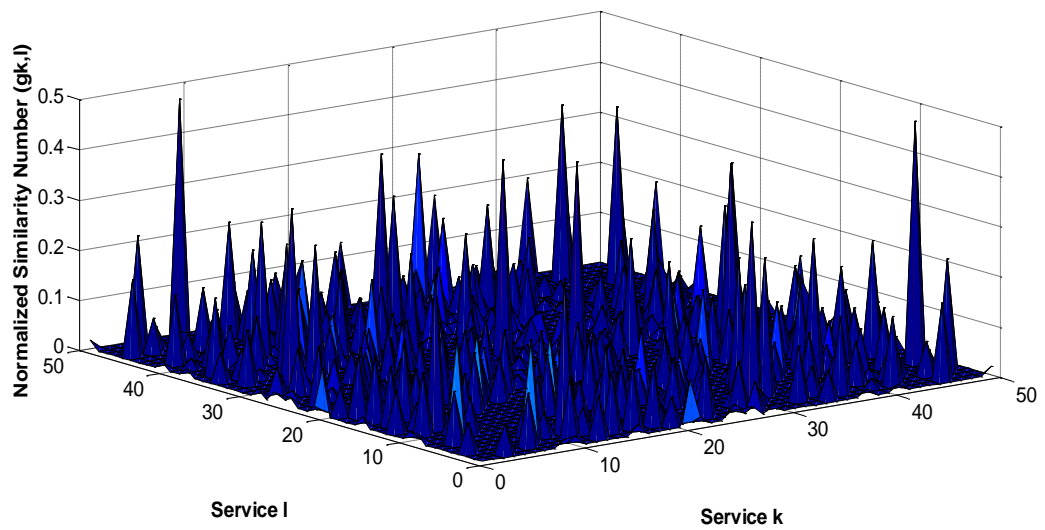
```

For each service  $s_k$  in 50 services
  Counter =0;
  For each keyword  $k_i$  in the set of keywords of  $s_k$ ,  $K_k$ 
    For each service  $l$ ,  $l \neq k$ 
      {
        For each keyword  $k_j$  in  $K_l$ 
          If  $k_i == k_j$ 
            Counter++;
      }
  Weight of similarity between  $s_k$  and  $s_l$ 
   $g_{k,l} = \text{counter}/\text{length}(s_k \cup s_l)$ 
  Counter = 0;

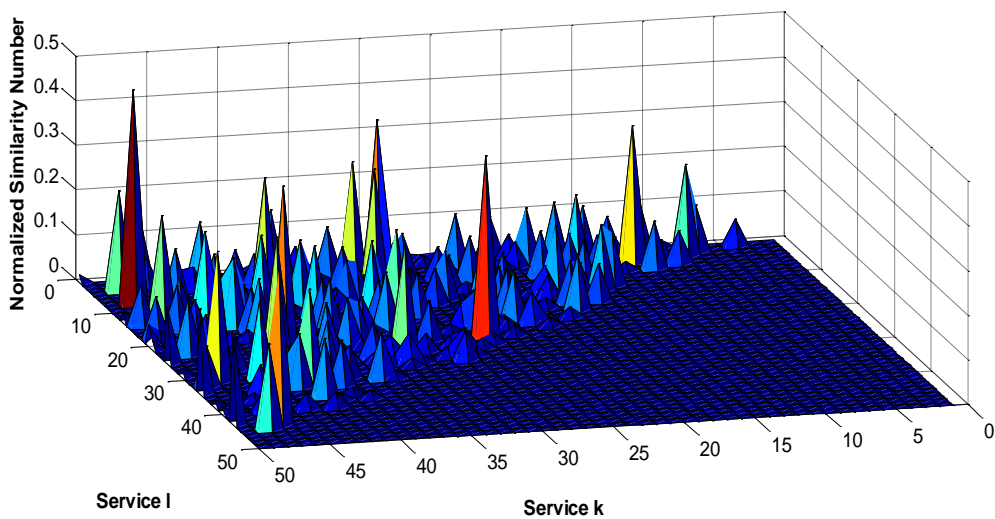
```

In order to visualize the distribution of this similarity matrix, a surface plot is drawn as shown in Figure 5.2. In (a) all the matrix is represented while in (b) only the upper triangle diagonal of the matrix is shown (knowing that it is symmetric as previously discussed). Based on this plot, we can see that the normalized similarity number varies between 0 and 0.4828. It is evident that the similarity number $g_{k,l}$ reaches a high value when the services belong to the same category and gets a low one when services belong to different ones. For instance, the similarity number for services 8 and 48 is high equal to 0.4828 since both services belong to the same category of applications: “Games”. However, $g_{k,l}$ is equal to 0 for services 1 and 49 since service 50 belongs to the “Traffic” category which is independent of the “School Tutoring” one, having no common keywords. Moreover, we calculated the average and standard deviation for this matrix and found to be 0.0788 and 0.0754 respectively, which will be used later. Note that the similarity numbers equal to 0 were excluded from this computation because they show the independency of two services belonging to different categories and the average similarity means the average of only those services that are similar.

In order to see the matrix from another perspective, we plotted it in terms of the number of common keywords (without normalization) between two similar services as seen in Figure 5.3. It is worthy to note that the average similarity and standard deviation were 3.4818 and 3.0227, respectively while the similarity number values range between 0 and 20.

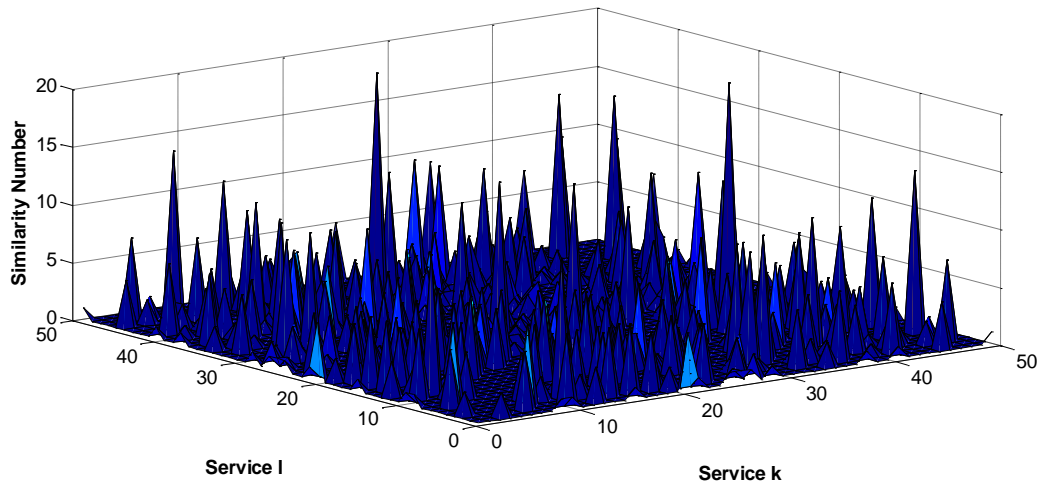


(a)

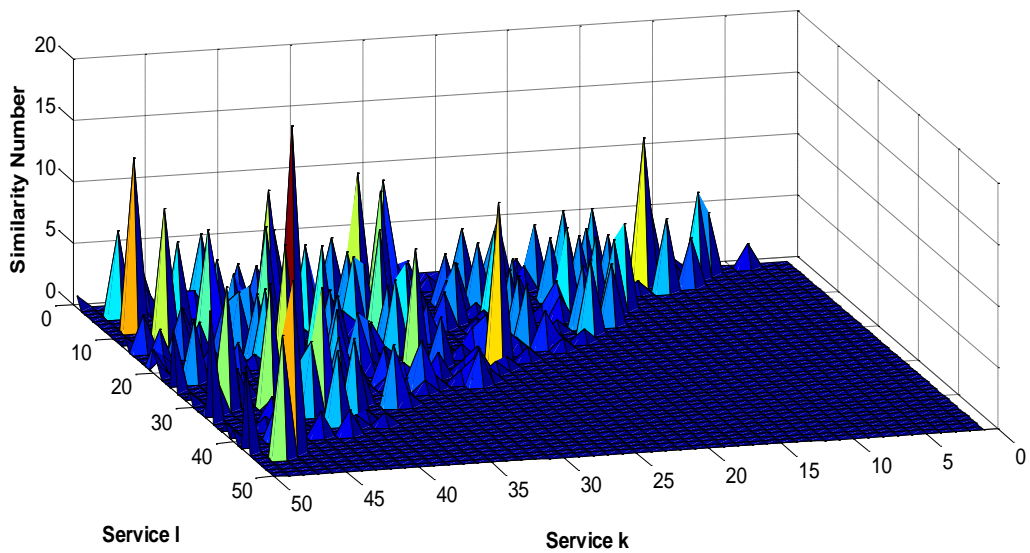


(b)

Figure 5.2 (a) Normalized similarity matrix (b) Upper triangle diagonal of the matrix



(a)



(b)

Figure 5.3 (a) Similarity Number Matrix (b) Upper triangle diagonal of the matrix

To simulate the requester's intent (i.e., ability of the requester to specify the desired service through a set of keywords), we concatenate the keywords of the services having a similarity number greater than a given threshold T which defines the requester's knowledge, i.e a high threshold means that the requester chooses targeted words while discovering a service. In the simulation, this will be translated by letting

the requester choose from a smaller pool which resulted from a less number of services' keywords aggregated. A $g_{k,l}$ equal to 0 induces that the service has no common keywords with the other one, therefore no additional keywords are appended to its own keywords (like in s_2 and s_{50} in Figure 5.4). If $g_{k,l} > 0$ and $g_{k,l} > T$, the keywords of services s_k and s_l will be aggregated (like in s_1, s_3, s_4). By this, the "concatenated" lists cl are created. Note that the aggregation of services having similar keywords will cause redundant words. However, the simulator will not delete these repeated words since this imitates the probability of mobile users selecting some words more than the others.

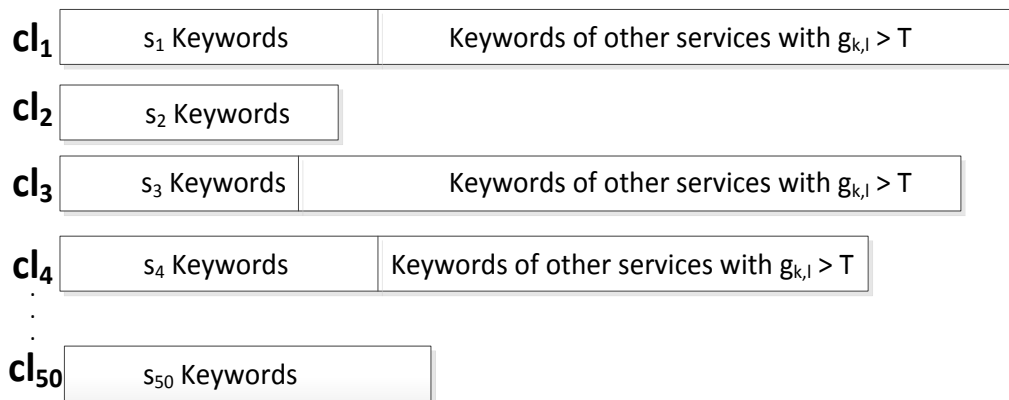


Figure 5.4 Concatenated list

In order to simulate the requester's intention in looking for a service, the simulator maps the index of the requester's randomly chosen service to one of the concatenated lists of services. Then, the requester chooses Q keywords ($2 \leq Q \leq 6$) from the list of concatenated sets of keywords as shown in Figure 5.5, which represents the concatenation list (Cl) of three sets of keywords belonging to three similar services (sr_1, sr_2, sr_3) where the intended service (sr_1) is one of them. The figure also illustrates that the user in this particular example had intended to select service sr_1 , and for this he specified keywords $y_1^{sr_1}, y_2^{sr_1}, y_1^{sr_2}, y_5^{sr_2}, y_{10}^{sr_2}, y_1^{sr_3}$. In this case, his request will result in

a “hit” (i.e., matching at least one keyword of sr_1), but we also observe a false positive, since a match also occurred with sr_2 and sr_3 . Note that the user could have chosen for example $y_2^{sr_2}, y_7^{sr_2}, y_2^{sr_3}$, in which case a “miss” will result.

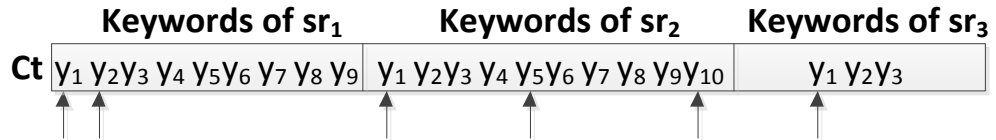
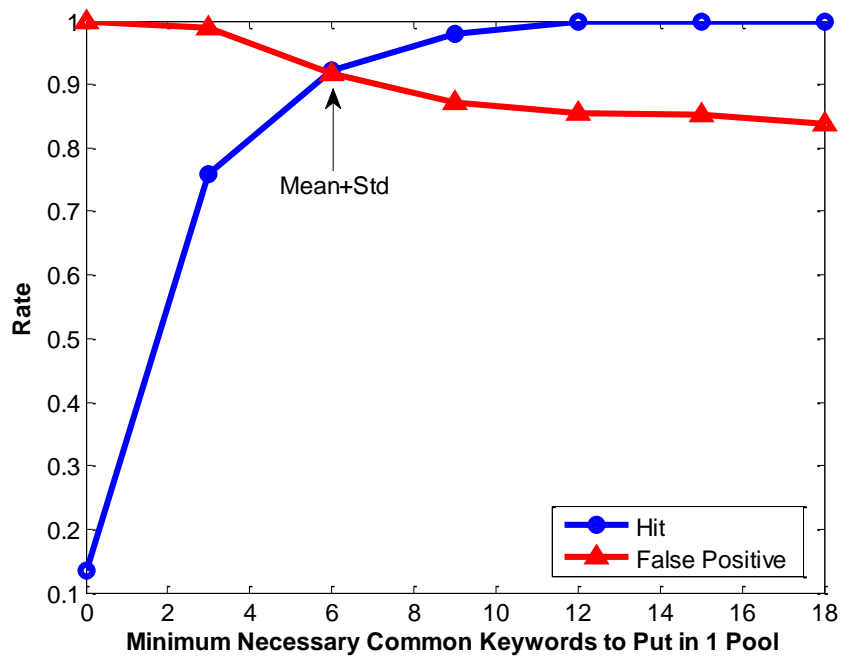


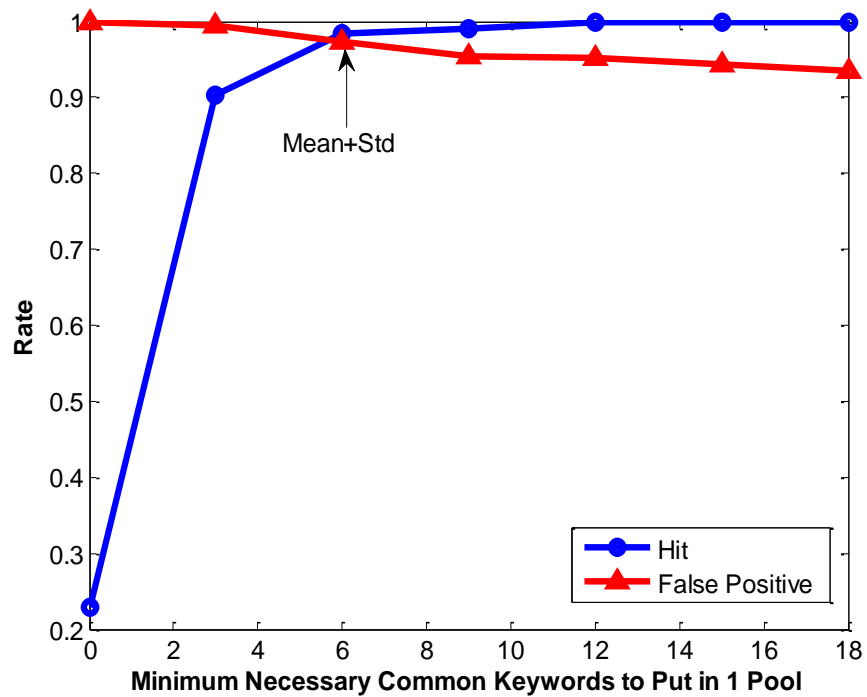
Figure 5.5 Requester’s choice from the concatenated service keywords list

It is obvious that the similarity threshold and the number of requester’s keywords are the parameters affecting the discovery performance on the ProSe Function. For this reason, we study the influence of their variations on the number of hits and false positives.

We start by testing the threshold’s effect (the number of common keywords between 2 services) on the system performance, and for this we vary the threshold within the range, starting with the (*mean – standard deviation*) and ending with the *maximum similarity number* (20) in steps of 1 *standard deviation* (*std*). In these simulations, which were repeated 1000 times, we fix the number of requester’s keywords to 3 and 5 respectively. Accordingly, we generate plots for the average number of hits and false positives versus the minimum necessary number of common keywords in order to put these services’ keywords in one pool, as shown in Figure 5.6 (a) and (b).



(a)



(b)

Figure 5.6 Average Number of hits and false positives versus minimum number of necessary common keywords to put in one pool (a) 3 keywords/request (b) 5 keywords/request.

As these figures show, the average number of hits grows exponentially with the increase of the threshold until it reaches a value of 1 after the threshold reaches a value corresponding to the (*mean + 3 standard deviations*) in the case of 3 keywords/request, Figure 5.6 (a) and (*mean + 2 standard deviations*) in the case of 5 keywords/request, Figure 5.6 (b). On the other hand, the number of false positives decreases with the increase of the threshold until it reaches a minimum of 0.88 for a threshold equal to (*mean + 4 standard deviations*) for the first case and a minimum of 0.98 for a threshold equal to (*mean + 2 standard deviations*) for the second case

It is good to mention also that the average number of false positives becomes less than the one of hits after a value of 6 common keywords in both cases (*mean + standard deviation*). We can say that these graphs illustrate the requester's precision in selecting his keywords. In other terms, a 0 value means no common keywords are needed in order to decide whether these services should be concatenated or no, therefore all the services will be in one pool. When the requester chooses keywords from this large pool, the probability of choosing his intended one is very low. That's when the requester is not specific in choosing his keywords; the average number of false positives is 1 while the average number of hits is very low equal to 0.25. However, when the necessary common keywords to put in one pool increases, fewer services fulfilling the criterion will be concatenated, and consequently, the requester will choose from a smaller pool, increasing by this the probability of choosing the targeted keywords, and hence the intended service. If we compare 5.6 (a) and (b), we can find that the average number of hits reaches a maximum of 1 for a less value for the threshold since the more the requester chooses keywords, the more is the probability of finding the indented service.

In the second experiment, in order to study the impact of the number of requester's keywords on the simulator's performance, we fix the threshold to (*mean + standard deviation*), and vary the requester's number of keywords between 1 and 6. We chose this particular value based on the results from the previous experiment, where the number of hits is greater than the number of false positives for this threshold value. As before, we plot the average number of hits and false positives versus the number of requester's keywords, as shown in Figure 5.7.

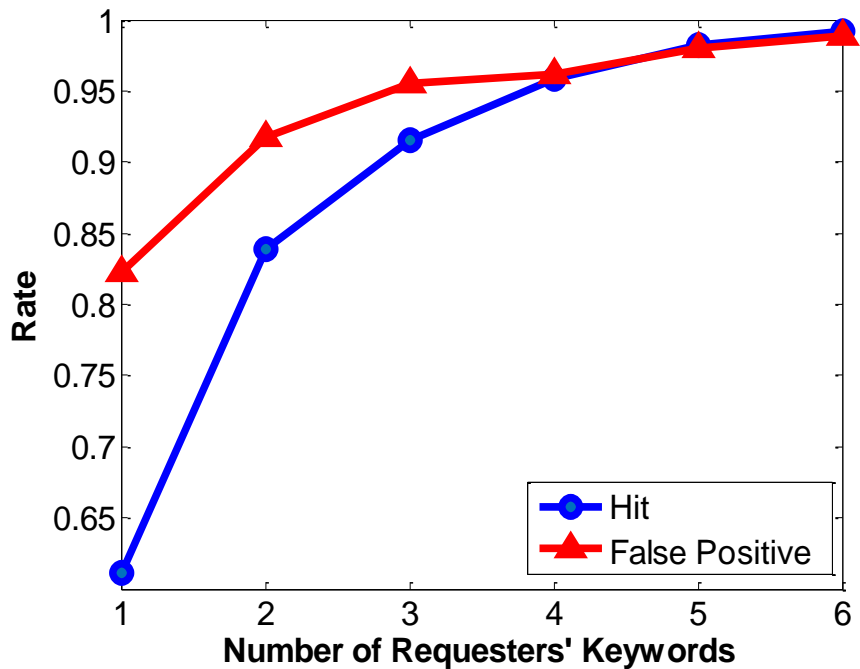
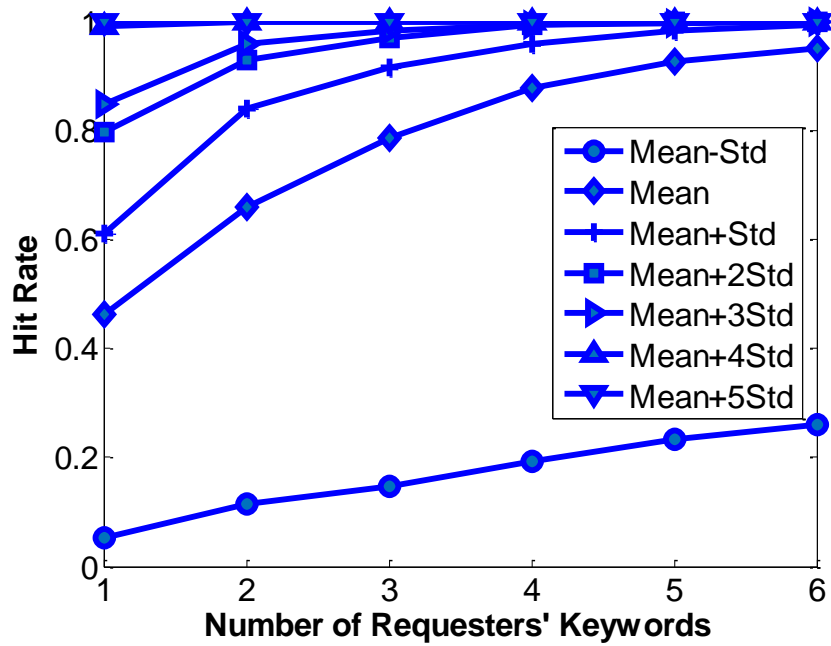


Figure 5.7 Average number of hits and false positives versus the number of requesters' keywords

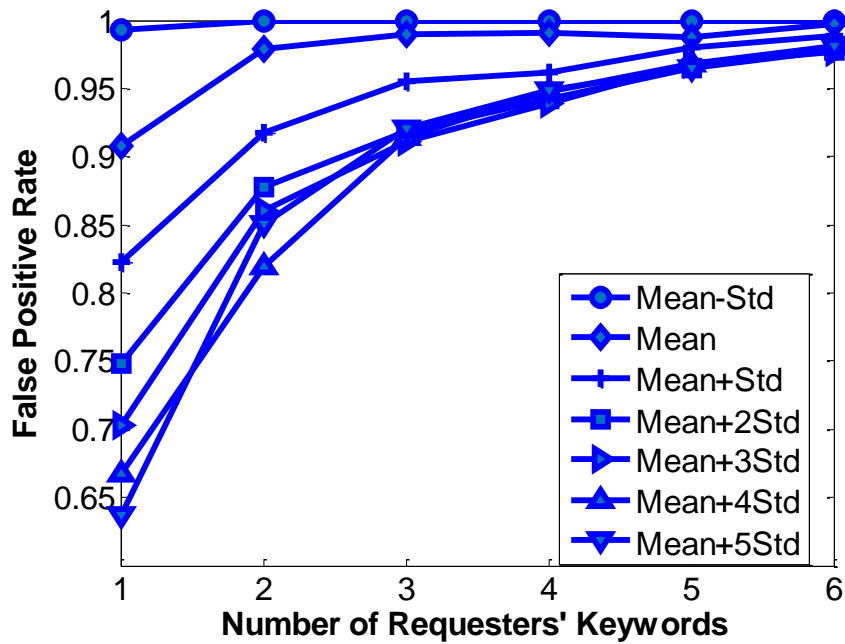
As the figure reveals, the average number of hits increases exponentially with the increase of number of keywords and reaches a maximum of 0.98 for 6 keywords. The number of false positives also increases with the increase of the number of keywords and reaches a value of 0.98 for 6 keywords. The increase in the average number of false positives can be attributed to the fact that when a requester chooses

more keywords, the probability of choosing from services other than the intended one becomes higher.

In order to test the requester's knowledge while entering the keywords, we plot the variation of the average number of hits and false positives versus the number of requester's keywords for different threshold values as shown in Figure 5.8 (a) and (b) respectively. These graph show that the average number of hits for the same number of keywords increases with the increase of the threshold used while the average number of false positives decreases. For instance, if the requester chooses 3 keywords, the average number of hits is less than 0.2 if the threshold considered is $(mean - std)$ while it is 1 if $T = mean + 4 std$. This simulates the precision of the requester while choosing targeted keywords for discovering the intended service. The explanation for this can be that a higher value for the threshold leads to less number of concatenated services since these services should have a high similarity number in order to have their keywords aggregated. In real life, some services are defined by exact keywords with no similar keywords with other services. For instance, a 'Game' service has specified keywords like "win", "player", "competition" that cannot be found in other services.



(a)



(b)

Figure 5.8 Average number of hits and false positives versus the number of requesters' keywords for different threshold values

B. Provider Discovery

1. *System Environment*

After testing the requester's intention, we study the simulator's performance in terms of helping the requester in discovering near providers offering his or her desired service. For this purpose, we consider a simulation environment of a 750 users moving in an area of $400 \times 400m^2$ for 1000 s. These nodes are divided into providers and requesters with a percentage of 6% of providers in the network. We assume that each provider holds, on his mobile phone, one service of the 50 services existing in the network as shown in Table 5.1. While registering it in the network, the provider chooses a random number of keywords (between 3 and 8) from the initial pool dictionary (Figure 5.2) in order to be used as tags to be discovered by. As for the requester, we assume that he or she is entering 3 keywords in the request (based on Figure 5.6) which are used to select from the concatenated list (Figure 5.4) that aggregates similar services having similarity number greater or equal to (*mean + standard deviation*) of the matrix. We assume 3 keywords / request and not 5 keywords/ request taking into consideration the requester's potential in entering keywords. As for mobility, we assume, on a hand, that the requesters are moving according to the Random Way Point mobility pattern since it imitates the human's mobility. On the other hand, we suppose that the providers are fixed at predefined positions since they should be more or less stable in order to stay connected with the requester and offer his or her service. By default, we assume that the requesters are pedestrians having a velocity between 0.1 and 3 meters per second, can pause for a time between 0 and 300s, walk in all directions (between -180 and 180 degrees) for duration between 30 and 500s. The results are reported by a time step of 10s. Note that a provider is considered to be in proximity of

the requester when the distance separating them is not more than 25 m [23] so that when the session between the provider and requester starts, we can guarantee that the communication stays held. Figure 5.9, extracted from this reference, justifies the choice of the maximum distance since the throughput between D2D pairs decreases when the distance between them increases, reaching a minimum of 2.5 Mb/s for 25 m.

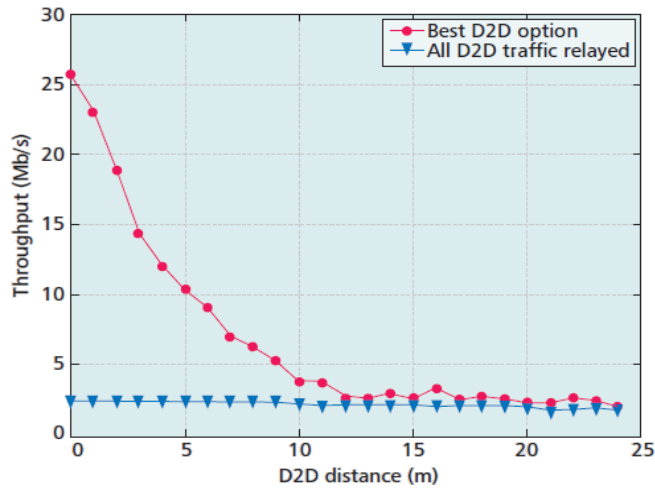


Figure 5.9 D2D throughput as a function of the D2D link distance [23].

The parameters' default values applied in our simulations are summarized in

Table 5.2.

Table 5.2 Default parameters

Parameter	Default Value
Simulation Time	1000 s
Total Number of UEs	750
Number of providers	6% of Total Number of UEs
Number of Services	50
Number of Services per Provider	1
Number of Providers' Keywords	Randi[3,8]
Number of Requester's Keywords	3
Threshold	(Mean + Std) of Similarity Number
Area (length, length)	(400×400) m ²
Maximum Distance between D2D pairs	25 m
Mobility Pattern	Random Way Point
Speed Interval	[0.1 3] m/s (Pedestrian)
Pause Interval	[0 300] s
Walk Interval	[30 500] s
Direction Interval	[-180 180] degrees
Time Step	10

2. System Performance

When the simulation starts running, the requesters move in an area of $(length \times length) m^2$ while the providers are fixed at predefined positions in an area of $(0.1 \times length) \times (0.9 \times length) m^2$. Figure 5.10 illustrates a prototype of 20 requesters ($R_1 \dots R_{20}$) and 5 providers ($P_1 \dots P_5$) marked in red, all distributed in the simulation area. We represent here that the providers' offering depends on the requesters' demand. For instance, P_1 emerges where there is a large number of requesters (R_3, R_4 and R_5) in busy areas like cities, downtowns, etc. However, R_{16} exists in a place far from the city and hence has no providers appear around it.

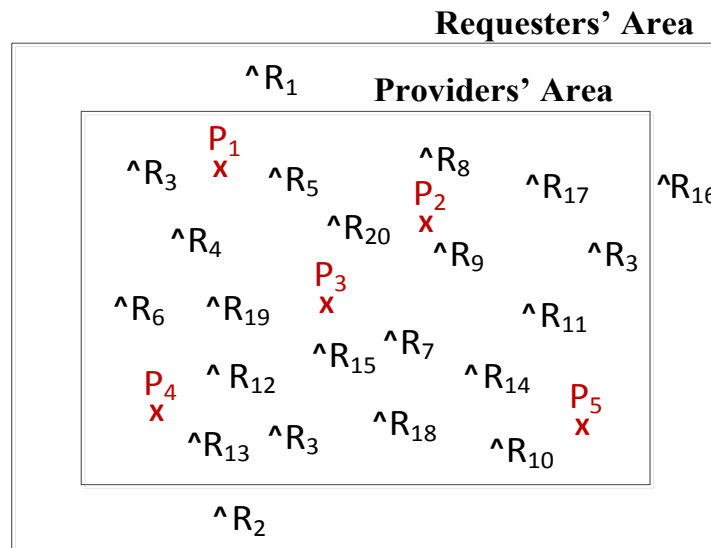


Figure 5.10 Providers' and requesters' areas

Each provider registers himself or herself in the network by choosing randomly one of the 50 services, from which he or she will choose a random number of keywords between 3 and 8. Figure 5.11 represents P providers (pv) choosing from 50 services a number of keywords (selected in gray). As an example, provider pv_1 chooses sp_1 as a service to share with requesters and picks 5 keywords out of its 9 keyword's pool.

Note that two providers can choose the same service as it is the case for pv_2 and pv_3 offering sp_2 as a service.



Figure 5.11 Provider's chosen service and keywords

At time t , a requester creates a request in order to search for a service by entering a number of keywords (according to the requester's intention previously discussed).

$$Y = \{y_Q\} = \{y_1^{sr_1}, y_2^{sr_1}, y_1^{sr_2}, y_5^{sr_2}, y_{10}^{sr_2}, y_1^{sr_3}\}$$

This request may contain keywords $y_1^{sr_2}, y_5^{sr_2}, y_{10}^{sr_2}, y_1^{sr_3}$ that belong to services sr_2 and sr_3 other than the intended one sr_1 as marked in bold in Figure 5.12.

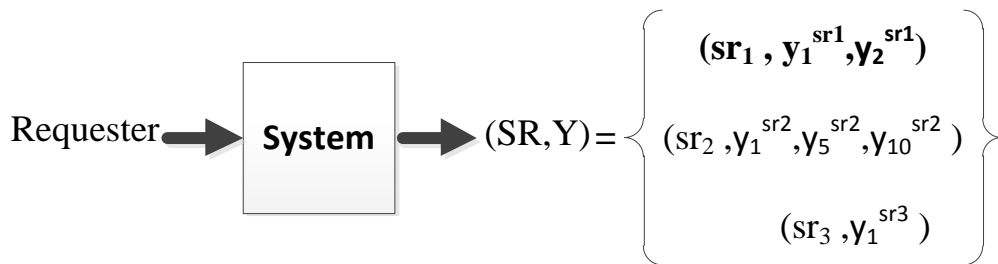


Figure 5.12 Requester's request creation

At the first stage, the simulator searches for the providers' ids offering the services matching with the entered keywords using also Minimum Edit Distance algorithm. In our example, the simulator looks up for sr_1, sr_2, sr_3 in its database illustrated in Figure 5.11. It finds that pv_1 offer sr_1 while pv_2 and pv_3 both propose sr_2

as service. Hence, pv_1, pv_2 and pv_3 are considered as matching providers and will be named as pm_1, pm_2 and pm_3 . It can happen that a service does not appear in the database; hence, no providers are offering it, as it is the case for sr_3 . At the next stage, the simulator compares the requester's keywords with the ones of the providers having these services and creates a list (Figure 5.13) containing the following information: the provider ID, the matching weight (the number of matches with the requester's keywords) and their indices. The matching index is the order of the providers' keyword relatively to the requesters' keywords present in the request. For instance, if we recall the requesters' keywords $\{y_1^{sr_1}, y_2^{sr_1}, y_1^{sr_2}, y_5^{sr_2}, y_{10}^{sr_2}, y_1^{sr_3}\}$, we find that provider pm_1 has 1 match with the requester's keywords while pm_3 has 3. As for indices, pm_1 has the 1st keyword $y_1^{sr_1}$ while pm_3 has the third, fourth and fifth $y_1^{sr_2}, y_5^{sr_2}, y_{10}^{sr_2}$.

Provider ID	Matching Weight	Matching Indices
pm_1	1	1
pm_3	3	3,4,5

Figure 5.13 List of providers matching keywords

However, even if pm_3 has a higher matching weight, he can be far away from the requester and thus he cannot establish a D2D session with him. For this reason, the distance between the requester and each provider in the list should be computed. We assume that the network is aware of the requester's position at time t and asks for the providers' positions. (i.e pm_1 and pm_3) in order to calculate the Euclidean distance between them. According to the Euclidean distance formula, the distance between two points in the plane with coordinates $(x_{requester}, y_{requester})$ and $(x_{provider}, y_{provider})$ is given by:

$$dist(requester, provider) = \sqrt{(x_{requester} - x_{provider})^2 + (y_{requester} - y_{provider})^2}$$

In order to consider a provider is in the requester's proximity, this distance should be less than 25m as aforementioned. The simulator filters the list in Figure 5.13 by removing all the providers having $dist > 25$. The new list will be as depicted in Figure 5.14 (b).

Provider ID	Matching Weight	Matching Indices	Distance
pm_1	1	1	18
pm_3	3	3,4,5	30

(a)

Provider ID	Matching Weight	Matching Indices	Distance
pm_1	1	1	18

(b)

Figure 5.14 (a) List of providers matching keywords along with their matching indices and distances (b) Filtered list

For instance, assume pm_3 is far from the requester by 30 m while pm_1 by 18m. Though pm_3 has a higher matching weight than pm_1 , he is further away from the requester and hence he should be discarded from the list that will be sent to the requester as shown in Figure 5.14 (b). Note that if this list contains more than one provider, the requester will compromise between the number of matches and distances for providers according to his utility. For example, if the list includes in addition to provider pm_1 , a provider pm_2 with a matching weight of 4 and a distance of 24 m, the requester will have the freedom to choose the nearest one pm_1 with less matching weight or the one having higher number of matching keywords pm_2 .

As we can see, the sources of no perfect matching (errors) between the request and the corresponding service on the simulator level can be summarized below:

- A requester not specifying all the keywords registered for the service in the network
- A provider not registering all the relevant keywords of his offered service
- A small number of providers in the network

For this reason, to test our system's performance, we consider the following metrics: the variation of number of providers in the network, the number of keywords the providers choose while registering themselves in the prose function's database and the number of keywords the requester inserts while discovering a service. In addition to this, we study the influence of the dimensions of the simulation area as well as the requester's speed on the number of providers discovered.

In the first experiment, we vary the percentage number of providers between 1% and 10% of the total number of UEs in the network and we repeat the simulations 1000 times. Note that the other simulation's parameters are assigned to the default values presented in Table 5.2. We plot the average number of discovered providers versus the number of providers in the network in Figure 5.15.

As the plot shows, the average number of discovered providers increases almost linearly with the increase of the number of providers in the network. It starts by an average of 0.25 discovered providers for 1% providers of the total number of UEs in the network then reaches a value of 2.4 for 10%. This can be explained by the fact that when the number of providers increases in the network, the probability of having providers matching the desired service by the requester increases.

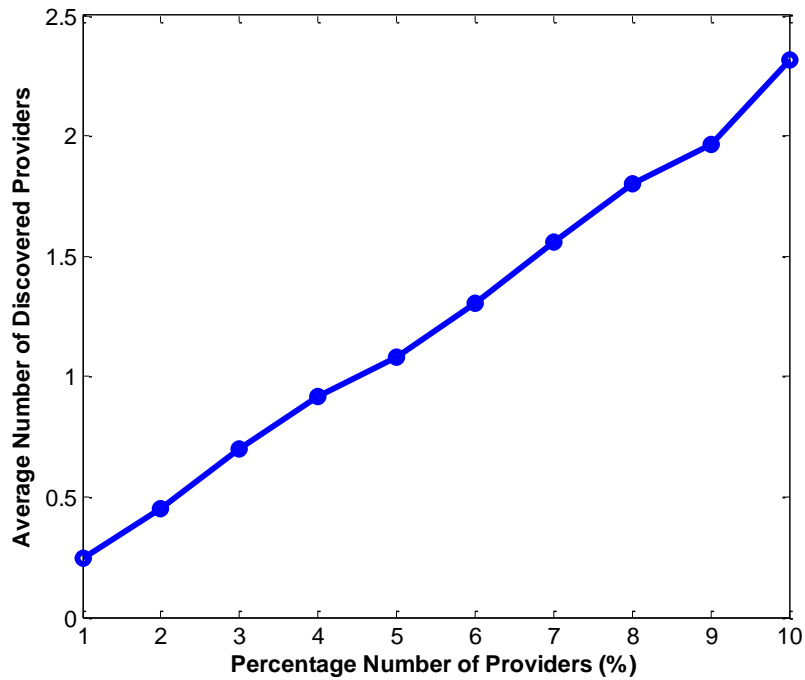


Figure 5.15 Average number of discovered providers versus the percentage number of providers in the network (regardless of proximity)

In the second experiment, we fix the number of providers to be 6% providers of the total number of UEs and change the number of keywords the providers choose while registering their service in the network between 3 and 8. All the values for the other parameters are according to Table 5.2. In Figure 5.16, we plot the average number of discovered providers versus the number of providers' keywords.

As the plot reveals, the average number of discovered providers grows with the increase of the number of providers' keywords. It begins by a small value of 0.79 for 3 keywords chosen by the provider and ends by much higher one: average of 1.8 discovered providers for 8 keywords. This points out that when the network's database contains more keywords per provider, the probability of finding him having more matching keywords with the requesters' keywords becomes higher.

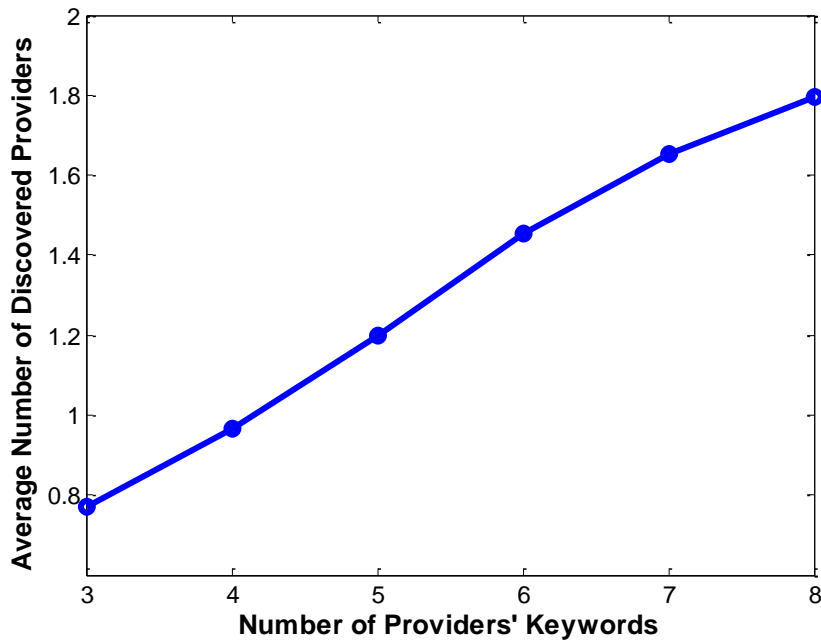


Figure 5.16 Average number of discovered providers versus the number of providers' keywords (regardless of proximity)

In the third experiment, we vary the number of the keywords inserted by the requester between 1 and 6. The other parameters are compliant with Table 5.2. We plot the average number of discovered providers versus number of requester's keywords in Figure 5.17.

As we can see in Figure 5.17, the average number of discovered providers rises with the number of requester's keywords' increment. It starts by an average of 0.5 for 1 keyword per requester and grows to reach 2.3 for 6 keywords. The analysis for this can be that the more the requester inserts keywords, the more he is giving information about his desired service, the more the simulator is certain while choosing the corresponding providers.

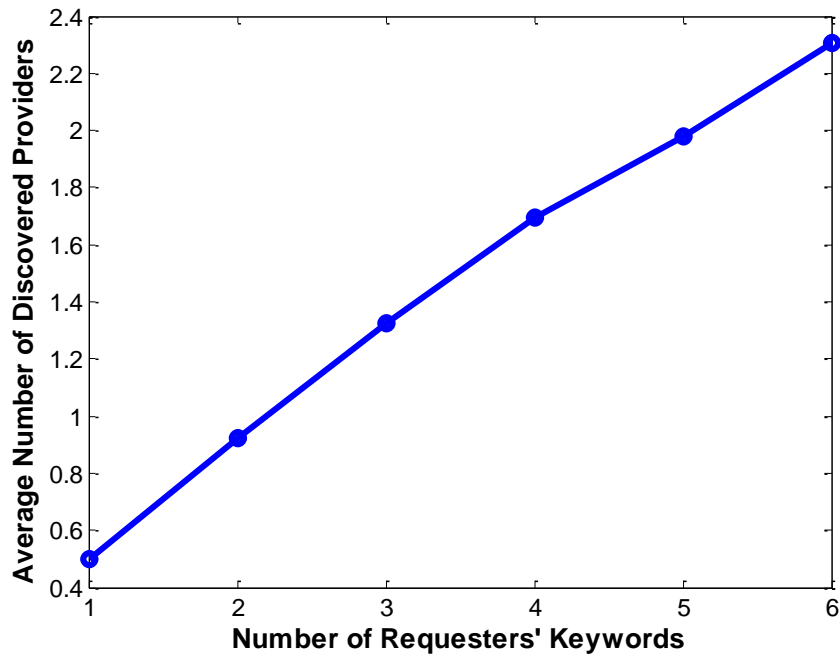


Figure 5.17 Average number of discovered providers versus the number of requesters' keywords (regardless of proximity)

In the next experiment, we vary the area of the simulation environment between $100 \times 100m^2$ and $1000 \times 1000m^2$. It is good to remind you here that the area the providers are enclosed in is a factor of the length of the total area (between $0.1 \times length$ and $0.9 \times length$). We consider here two scenarios for the requesters moving in a Random Way Point mobility pattern: in the first one, the requesters are pedestrians moving according to the parameters defined in Table 5.2, while in the second scenario, the requesters are driving vehicles in a speed between 12 and 24 m/s for a duration ranging between 30 and 300s and can pause between 0 and 120s. We plot the average number of discovered providers versus the length of the simulation area for the two scenarios in Figure 5.18.

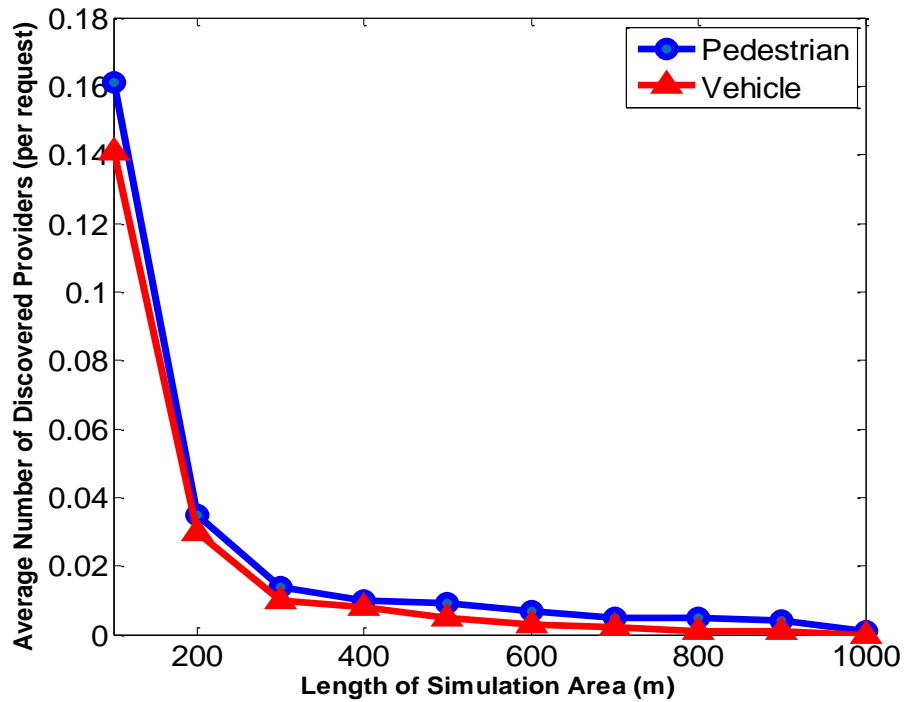
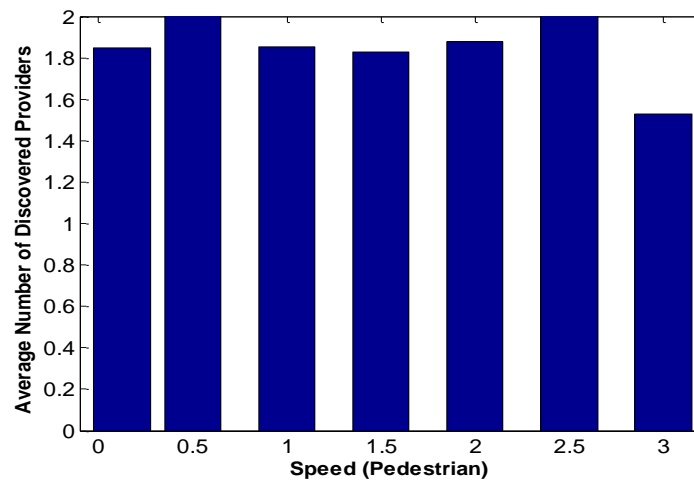


Figure 5.18 Average number of discovered providers (per request) in requester's proximity versus the length of the simulation's area.

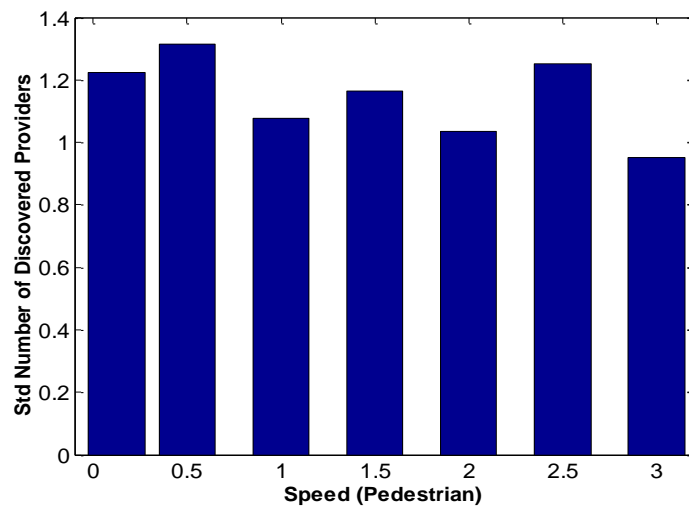
As the plots present, the average number of discovered providers decreases when the area increases in both scenarios: when the requester is a pedestrian or driving a vehicle. In pedestrian case, the average number starts by 0.16 for an area of $100 \times 100m^2$ until it reaches around 0 for $1000 \times 1000m^2$. As for vehicle, it starts by 0.14 for 100m until it reaches around 0 for 1000m. This can be analyzed by the fact that when the area is small and condensed with providers, the probability of finding a nearby provider matching with the desired service increases.

In the last experiment, we plot (a) the average number and (b) the standard deviation of discovered providers versus the requester's speed for both scenarios: walking requester (Figure 5.19) and driving vehicle (Figure 5.20). In our representations, we omit the case of pauses times. In both scenarios, we consider that the providers are fixed at predefined positions, choosing the same services in the 1000

runs and selecting 5 keywords from the service’s pool. The requester is still choosing 3 keywords each time. Note that the maximum distance to consider a provider is in the requester’s proximity is considered to be equal to 25. Obviously, the average number of discovered providers is not the same at all speeds since the requester may get near of some providers but at the same time get away from others. Hence, an important observation is that the distribution is independent of the speed of the requesters.

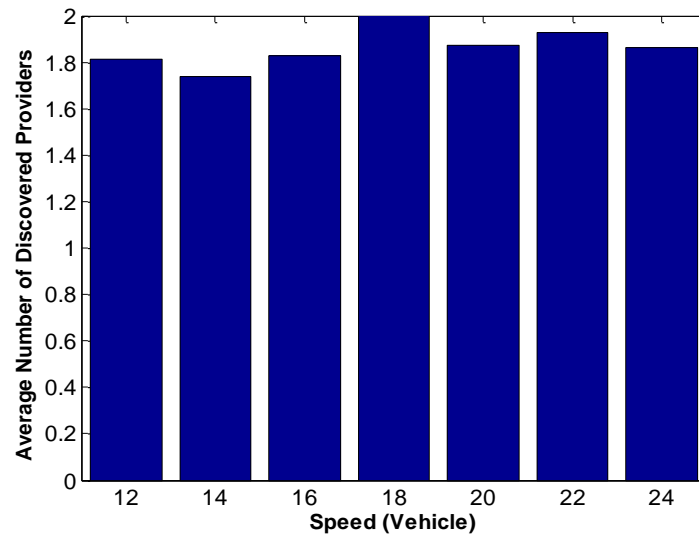


(a)

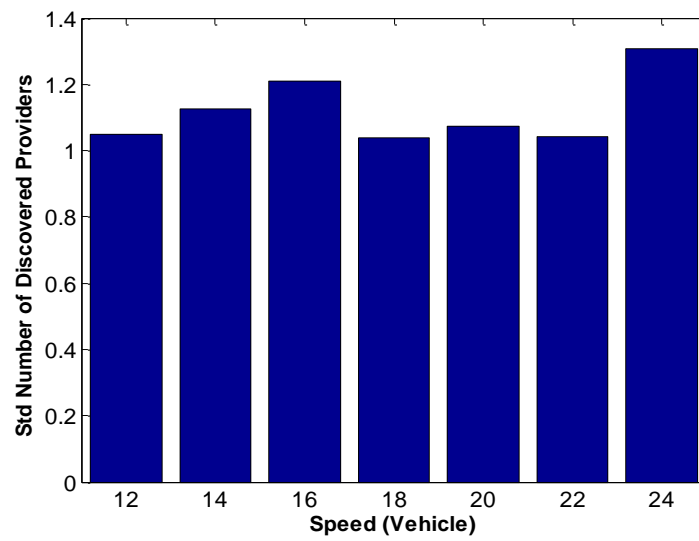


(b)

Figure 5.19 (a) the average number and (b) the standard deviation of discovered providers versus the requester’s speed (Pedestrian scenario, $D_{\max}=25\text{m}$).



(a)



(b)

Figure 5.20 (a) the average number and (b) the standard deviation of discovered providers versus the requester's speed (Vehicle scenario, $D_{\max}=25\text{m}$).

CHAPTER VI

CONCLUSION AND FUTURE WORK

A. Conclusion

In this work, we presented a general design of a cloudlet-inspired D2D scheme, where a requester asks for an application through keywords, leaving it up to the network to locate devices in proximity that offer the requested services. Our proposed design was meant to extend the application services usually used in D2D communication by integrating mobile cloud services into this technology: Software as a Service (SaaS), Infrastructure as a Service (IaaS) and Platform as a Service (PaaS). The first type of services was addressed in our study whereas IaaS and PaaS will be the subject of future work. Our proposed work aimed also at designing a system for D2D communication (with Software as a Service) that fits the LTE architecture and abides by the relevant standards, showing all the enhancements to the network core (architectural elements, elements design, signaling, and functions) that are needed for service registration and peer discovery. Finally, it should be noted that the proposed work took the Third Generation Partnership Project (3GPP) Long Term Evolution (LTE) system as a baseline.

Moreover, we have modeled the requester's intention while entering his keywords searching for a certain application by testing his ability to specify his desired service through a set of keywords. We found that the number of these keywords entered affects the probability of hit and false positives (number of candidate providers, having matching keywords, sent back by the network). We defined a threshold that induces the

requesters' knowledge about this service since many applications belonging to the same category may have common keywords. A high threshold induces that the requester is using targeted keywords in his request.

Furthermore, we tested analytically and experimentally many parameters affecting the system performance. The sources of no perfect matching (errors) between the request and the corresponding service on the simulator level are 1) a requester not specifying all the keywords registered for the service in the network 2) a provider not registering all the relevant keywords of his offered service 3) a small number of providers in the network. For this reason, to test our system's performance, we considered the following metrics: 1) the number of keywords the requester inserts while discovering a service 2) the number of keywords the providers choose while registering themselves in the prose function's database 3) the variation of number of providers in the network. In addition to this, we study the influence of the dimensions of the simulation area as well as the requester's speed on the number of providers discovered. Experiments showed that 1) the more the requester inserts keywords, the more he is giving information about his desired service, the more the simulator is certain while choosing the corresponding providers 2) when the network's database contains more keywords per provider, the probability of finding him having more matching keywords with the requesters' keywords becomes higher 3) when the number of providers increases in the network, the probability of having providers matching the desired service by the requester increases. Results also showed that when the simulation area is small and condensed with providers, the probability of finding a nearby provider matching with the desired service increases. Besides, an important observation was that

the distribution is independent of the speed of the requesters since the requester may get near of some providers but at the same time get away from others.

B. Future Work

For future work, we will explore the remaining cloud services namely Network as a Service (NaaS). This service may need additional signaling messages and functionalities for the existing nodes in the LTE-A network knowing that this service turns the provider's mobile to a hotspot WiFi Access Point for other devices that do not have access to the Internet.

Furthermore, charging in ProSe is also a subject to be tackled in our future work. We need to examine the required functionalities in the core network to collect charging data from the requester to ensure that users are charged for the resources they demand and use. The providers also need a consistent and predictable way to measure usage in order to benefit from the amount of resources they are renting. Note that we need to propose some incentives for the providers to lend their extra resources.

Moreover, in our scheme, we have considered that UEs are registered to same or different Public Land Mobile Networks (PLMNs) but without considering the scenario when the UEs are roaming. However, this should be examined in our future work since additional entities should be addressed (Home ProSe Function and Visited ProSe Function) along with the reference point between them in order to control the authorization and configuration of the UE for discovery. The location reporting is also an issue in this scenario since it has to be transferred between networks in order to determine the proximity of the UEs.

Besides, our proposed scheme consider only exact keywords search to retrieve applications of interest. So in future work, we will extend our scheme by adding the concept of fuzzy keyword search [55] on a hand and Natural Language Processing (NLP) on the other hand. First, the fuzzy keyword search would enhance system usability by returning the matching files when users' searching inputs exactly match the predefined keywords or the closest possible matching files based on keyword similarity semantics, when exact match fails. As for the NLP, which is a field of computer science, artificial intelligence, and linguistics concerned with the interactions between computers and human, we can turn the search more interactive by letting the requester enter his keywords by speaking to his phone. It is good to mention here, that in the case of fuzzy keyword search, we will still use edit distance k to quantify keywords similarity and assume a predefined edit distance d against which the result is compared ($k \leq d$). We will construct fuzzy keyword sets that incorporate not only the exact keywords but also the ones differing slightly due to minor typos, format inconsistencies, etc.

Finally, one closely related issue that must also be considered in any future work is the privacy protection requirements. The most important problem to dig into is how can the network revoke authorization and prevent malicious UEs from using ProSe capabilities.

BIBLIOGRAPHY

- [1] K. David, S. Dixit, and N. Jefferies, "2020 Vision The Wireless World Research Forum Looks to the Future," *IEEE Vehicular Technology Magazine*, vol. 5, no. 3, pp. 22–29, September 2010.
- [2] K. Samdanis, T. Taleb, and S. Schmid, "Traffic offload enhancements for eUTRAN," *Commun. Surveys Tuts.*, vol. 14, no. 3, pp. 884–896, 2012.
- [3] 3GPP, "3GPP TSG services and system aspects: Local IP access and selected IP traffic offload (LIPA-SIPTO) (Release 10)," 3GPP, Tech. Rep. TR 23.829, Aug. 2011.
- [4] Yang, Mi Jeong, Soon Yong Lim, Hong Joon Park, and Nam Hoon Park. "Solving the data overload: Device-to-device bearer control architecture for cellular data offloading." *IEEE Vehicular Technology Magazine* 8, no. 1 (2013).
- [5] 3GPP, "3GPP TSG SA: Feasibility Study for Proximity Services (ProSe) (Release 12)," 3GPP, Tech. Rep. TR 22.803, Aug. 2012.
- [6] K. Doppler and M. Xiao, Eds., "Innovative Concepts in Peer-to-Peer and Network Coding," WINNER+/CELTIC Deliverable CELTIC/CP5-026 D1.3, 2008.
- [7] IEEE Std. 802.11, "IEEE Standard for Information Technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications 1999 reaff 2007," 2007.
- [8] "Bluetooth," <http://www.bluetooth.com>.
- [9] P. J'anis, C.-H. Yu, K. Doppler, C. Ribeiro, C. Wijting, K. Hugl, O. Tirkkonen, and V. Koivunen, "Device-to-Device Communication Underlying Cellular Communications systems," *International Journal of Communications, Network and System Sciences*, vol. 2, no. 3, pp. 169–178, 2009.
- [10] G. Fodor, E. Dahlman, G. Mildh, S. Parkvall, N. Reider, G. Mikl'os, and Z. Tur'anyi, "Design aspects of network assisted device-to-device communications," *IEEE Communications Magazine*, vol. 50, no. 3, pp.170–177, March 2012.
- [11] L. Lei, Z. Zhong, C. Lin, and X. Shen. "Operator controlled device-to-device communications in LTE-advanced networks." *IEEE Wireless Communications*, no. 3 (2012): 96-104.
- [12] J. Rawadi, H. Artail, H. Safa, "Providing Local Cloud Services to Mobile Devices with Inter-cloudlet Communication", 17th IEEE Mediterranean Electrotechnical Conference (MELECON2014), Beirut, Lebanon, April, 2014.

- [13] M. Satyanarayanan, P. Bahl, R. Cáceres and N. Davies. “The Case for VM-based Cloudlets in Mobile Computing”, *IEEE Pervasive Computing*, v. 8, n. 4, 2009.
- [14] M. Satyanarayanan, “Mobile computing: the next decade”, *Proc. 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond (MCS '10)*, 2010.
- [15] D. Bernstein and D. Vij. “Intercloud Directory and Exchange Protocol Detail using XMPP and RDF”, *Proc. IEEE 6th World Congress on Services*, 2010.
- [16] E. Miluzzo, R. Cáceres and Y. Chen. “Vision: mClouds – Computing on Clouds of Mobile Devices”, *International Workshop on Mobile Cloud Computing and Services (MCS '12) with MobiSys 2012*, 2012.
- [17] C.-H. Yu et al., “Power Optimization of Device-to-Device Communication Underlying Cellular Communication,” *IEEE Int’l. Conf. Commun., ICC*, Dresden, Germany, June 2009.
- [18] J. P. Jänis et al., “Interference-Avoiding MIMO Schemes for Device-to-Device Radio Underlying Cellular Networks,” *IEEE Pers. Indoor and Mobile Radio Commun. Symp.*, Tokyo, Japan, Sept. 2009.
- [19] K. Doppler et al., “Mode Selection for Device-to-Device Communication Underlying an LTE-Advanced Network,” *IEEE Wireless Commun. and Networking Conf.*, Sydney, Australia, Apr. 2010.
- [20] J. Du, W. Zhu, J. Xu, Z. Li, and H. Wang, “A compressed HARQ feedback for device-to-device multicast communications,” in *Proceedings of IEEE VTC-Fall*, 2012, pp. 1–5.
- [21] B. Zhou, H. Hu, S.-Q. Huang, and H.-H. Chen, “Intracluster device-to-device relay algorithm with optimal resource utilization,” *IEEE Transactions on Vehicular Technology*, vol. 62, no. 5, pp. 2315–2326, Jun. 2013.
- [22] L. Lei, Z. Zhong, C. Lin, and X. Shen, “Operator controlled device-to-device communications in LTE-advanced networks,” *IEEE Wireless Communications*, vol. 19, no. 3, pp. 96–104, 2012.
- [23] K. Doppler, M. Rinne, C. Wijting, C. Ribeiro, and K. Hugl, “Device-to-device communication as an underlay to LTE-advanced networks,” *IEEE Communications Magazine*, vol. 47, no. 12, pp. 42–49, 2009.
- [24] N. Golrezaei, A. F. Molisch, and A. G. Dimakis, “Base-station assisted device-to-device communications for high-throughput wireless video networks,” in *Proceedings of IEEE ICC*, 2012, pp. 7077–7081.
- [25] J. C. Li, M. Lei, and F. Gao, “Device-to-device (D2D) communication in MU-MIMO cellular networks,” in *Proceedings of IEEE GLOBECOM*, 2012, pp. 3583–3587.

- [26] N. K. Pratas and P. Popovski, "Low-rate machine-type communication via wireless device-to-device (D2D) links," arXiv preprint arXiv:1305.6783, 2013.
- [27] X. Bao, U. Lee, I. Rimać, and R. R. Choudhury, "DataSpotting: offloading cellular traffic via managed device-to-device data transfer at data spots," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 14, no. 3, pp. 37–39, 2010.
- [28] E. Dahlman, S. Parkvall, and J. Sköld, *4G: LTE/LTEAdvanced for Mobile Broadband*, Academic Press, ISBN: 012385489X, 2011.
- [29] K. Doppler, C. B. Ribeiro, and J. Knecht, "Advances in D2D communications: Energy efficient service and device discovery radio," in *Proc. International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE)*. IEEE, Feb. 2011, pp. 1–6.
- [30] M. Scott Corson et al., "Toward Proximity-Aware Internetworking," IEEE Wireless Communication, Dec. 2010, pp. 26–33.
- [31] <http://www.qualcomm.com/solutions/wirelessnetworks/technologies/lte/lte-direct>.
- [32] 3GPP TR 23.703, "Study on architecture enhancements to support Proximity Services (ProSe) (Release 12)", v. 1.0.0, December, 2013.
- [33] Raghothaman, Balaji, Eric Deng, Ravikumar Pragada, Gregory Sternberg, Tao Deng, and Kiran Vanganuru. "Architecture and protocols for LTE-based device to device communication." In *Computing, Networking and Communications (ICNC), 2013 International Conference on*, pp. 895-899. IEEE, 2013.
- [34] 3GPP TR 33.833, "Study on security issues to support Proximity Services (Release 12)", v. 0.4.0, February 2014.
- [35] 3GPP TS 23.303, "Proximity-based services (ProSe); Stage 2 (Release 12), v.12.0.0, February 2014.
- [36] 3GPP TS 22.278, "Service requirements for the Evolved Packet System (EPS) (Release 12)", v. 12.4.0, September, 2013.
- [37] 3GPP TS 22.115, "Service aspects; Charging and Billing (Release 13)", v.13.0.0, December, 2013.
- [38] 3GPP TR 36.843, "Study on LTE device to device proximity services-Radio aspects (Release 12)", v. 1.0.0, November, 2013.
- [39] Open Mobile Alliance, OMA AD SUPL: "Secure User Plane Location Architecture", (<http://www.openmobilealliance.org>).
- [40] <http://tools.ietf.org/html/rfc3588#page-79>.
- [41] <http://tools.ietf.org/html/rfc2486>.

- [42] <http://tools.ietf.org/html/rfc821>.
- [43] 3GPP TS 23.032, "Universal Geographical Area Description (GAD) (Release 11)", v.11.0.0, September, 2012.
- [44] 3GPP TS 29.229, "Cx and Dx interfaces based on the Diameter protocol, Protocol details (Release 12), v12.3.0, September, 2014.
- [45] 3GPP TS 29.344, "Proximity-Services (ProSe) Function to Home Subscriber Server (HSS) aspects; Stage 3, (Release 12)", v0.2.0, May 2014.
- [46] 3GPP TS 29.345, "Inter-Proximity-Services (ProSe) Function signaling aspects; Stage 3, (Release 12), v0.2.0, June 2014.
- [47] 3GPP TS 29.272, "Mobility Management Entity (MME) and Serving GPRS Support Node (SGSN) related interfaces based on Diameter protocol (Release 12)", v.12.5.0, June 2014.
- [48] <http://probabilityandstats.wordpress.com/2010/02/18/the-matching-problem/>
- [49] C. Bettstetter, H. Hartenstein, and X. Pérez-Costa. "Stochastic properties of the random waypoint mobility model: epoch length, direction distribution, and cell change rate." In Proceedings of the 5th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems, pp. 7-14. ACM, 2002.
- [50] C. Bettstetter, H. Hartenstein, and X. Pérez-Costa. "Stochastic properties of the random waypoint mobility model." *Wireless Networks* 10, no. 5 (2004): 555-567.
- [51] A. A. Moffat and T. C. Bell, "Managing gigabytes: compressing and indexing documents and images", Morgan Kaufmann, (1999).
- [52] SensorTower website, www.sensortower.com
- [53] Considerations for server-side Automation of Office, Available. [online]. <http://support.microsoft.com/kb/257757>
- [54] [http://heuristicswiki.wikispaces.com/Levenshtein+distance+\(Heuristic\)](http://heuristicswiki.wikispaces.com/Levenshtein+distance+(Heuristic)).
- [55] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou. "Fuzzy keyword search over encrypted data in cloud computing." In INFOCOM, 2010 Proceedings IEEE, pp. 1-5. IEEE, 2010.