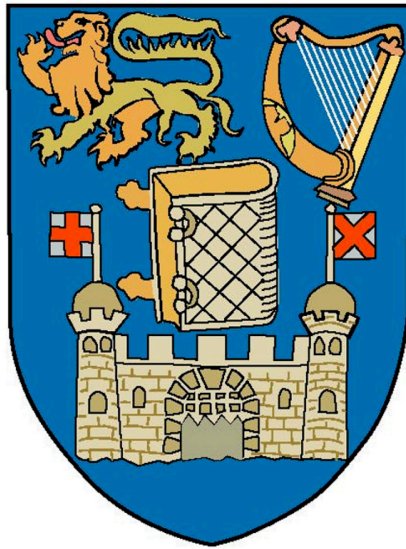




THE UNIVERSITY OF DUBLIN, TRINITY COLLEGE

On the Virtualization of Wireless Links

Author:

Jonathan VAN DE BELT

Supervisor:

Prof. Linda E. DOYLE

This thesis is submitted for the degree of
Doctor of Philosophy

8th June 2018

Declaration

I declare that this thesis has not been submitted as an exercise for a degree at this or any other university and is entirely my own work.

I agree to deposit this thesis in the University's open access institutional repository or allow the Library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

Signed:

Jonathan van de Belt,
8th June 2018

Summary

Virtualization is seen as a crucial component of future communication networks, whether wired or wireless, as it promises to deliver new services, increased flexibility, better resource efficiency, and faster technology development than is currently the case today. However, despite the prevalence of work on wired network virtualization, relatively little work has examined wireless network virtualization, although this area is now gaining increased attention.

One obstacle to research on virtualization is the ambiguous and vague manner in which the term ‘virtualization’ is employed, and the lack of a precise, technical definition. As a result, it is difficult to categorise the work on virtualization to date, and to determine meaningful research directions that can lead to the promised benefits of virtualization.

The first contribution of this thesis is to address the lack of clarity surrounding virtualization, by proposing a theory of virtualization which provides a rigorous framework for all aspects of virtualization. Additionally, the theory decomposes virtualization into two constituent parts: the isolation problem, and the embedding problem, and this provides many benefits, as will be shown. The theory is used to examine virtualization technologies in communication networks, and in particular wireless link virtualization.

Armed with a clear understanding of virtualization, the second contribution of the thesis is to perform a survey of existing wireless link virtualization techniques to discover relevant research topics. The theory provides a means of distinguishing virtualization techniques from those that are not, and enables structured classification of work done to date. As a result of this systematic examination, many new insights are reached, such as the realization that most research has focussed on the embedding problem without considering isolation, and that a form of virtualization called spectrum-level virtualization has largely been overlooked.

Following from the above observation, the third contribution of the thesis is to investigate the isolation problem for spectrum-level virtualization, both theoretically and practically. Wireless link virtualization is implemented on the CONNECT testbed, and the implementation is used to examine the isolation problem in real world conditions. Subsequently, applications of wireless link virtualization are proposed and demonstrated

on the testbed.

The final contribution of the thesis is to investigate the embedding problem at the spectrum-level in conjunction with the isolation problem. It is observed that to date embedding techniques have been static, and therefore, dynamic embedding algorithms are proposed. The benefits of dynamicity for wireless link virtualization are shown through simulations. Additionally, dynamic embedding is analysed in the context of a virtual public safety operator coexisting on a commercial wireless network through virtualization.

Finally, the merits of the approach taken in this thesis are discussed, and interesting future research is described.

Acknowledgements

As this chapter of my life draws to a close, I would like to thank all of the many people who have helped me along the way. Firstly, I would like to express my sincere gratitude to my supervisor Prof. Linda Doyle for her continued guidance, encouragement, insights, and enthusiasm over the past years. Secondly, thank you to Dr. Hamed Ahmadi for working closely with me, teaching me many valuable lessons, and always pushing me on when I needed it. Thank you also to the many people who have helped me academically, both through direct collaboration and through in-depth discussions. In particular, thank you to Prof. Oriol Sallent for his help with applications of my research, and to Dr. Johann Marquez-Barja and Dr. Nicholas Kaminski for kick-starting the development of the theory that underpins this thesis.

To all of the many members of CONNECT, past and present, and especially the occupants of the Ph.D. Lab, thanks for creating such an enjoyable and supportive work environment. It really is a unique place to work and I must thank Linda and Luiz for the existence of such a good atmosphere in the group. I am very grateful to share this office with such great people, Jacek, Danny, Arman, Sèamas, Pedro, Francisco, Carlo, Lele, George, Nima, João, Jernej, Boris, and so many others. Thanks for all of the fun times and the great memories. Thanks also to all of my friends who kept me going when I needed distractions outside of research, in particular my friends from music.

A huge thank you to my family, my parents Geert and Annie and my sisters Cecilia and Beatrijs, for their love and encouragement throughout many years. Most importantly of all, I would like to thank Eileen, without whom this thesis would have been impossible, for her unfailing love, support, and patience.

Contents

Declaration	i
Summary	iii
Acknowledgements	v
List of Figures	xiii
List of Tables	xvii
1 Introduction	1
1.1 Introduction	1
1.2 Virtualization of Communication Networks	1
1.3 The Many Meanings of Virtual	2
1.4 Technical Use of Virtualization	4
1.5 Objectives and Research Questions	6
1.5.1 Developing a Precise Understanding of Virtualization	6
1.5.2 Using the Precise Understanding of Virtualization	7
1.6 Key Contributions	8
1.6.1 Developing a Precise Definition of Virtualization	8
1.6.2 Using the Precise Definition of Virtualization	9
1.7 Thesis Outline	9
1.8 Publications	11
2 Prerequisites to Virtualization	13
2.1 Introduction	13
2.2 Abstraction/Representation Theory	14
2.2.1 Representation	14
2.2.2 Physical and Abstract Evolution	16
2.2.3 Instantiation	16
2.2.4 The Compute Cycle	18
2.3 Levels of Representation	18

2.4	Choosing a Hierarchy of Representation Levels	20
2.5	Hardware and Software through Representation	21
2.6	How Representation applies to Virtualization	22
2.7	Summary	23
3	A Theory of Virtualization	25
3.1	Introduction	25
3.2	Virtualization as Resource Mapping	25
3.3	Recursion	29
3.4	The Isolation Problem	32
3.5	The Embedding Problem	34
3.6	Definition of Virtualization	35
3.7	Validity of the Theory	35
3.8	Summary	37
4	Virtualization in Wireless Networks	39
4.1	Introduction	39
4.2	Nodes	39
4.2.1	Process Virtualization	41
4.2.2	Storage Virtualization	42
4.2.3	Machine (Node) Virtualization	42
4.2.4	Input and Output Virtualization	43
4.3	Links	43
4.3.1	Wired Link Virtualization	43
4.3.2	Wireless Link Virtualization	45
4.4	Wired and Wireless Network Virtualization	47
4.4.1	Wired Network Virtualization	47
4.4.2	Wireless Network Virtualization	52
4.5	Summary	55
5	Survey of Wireless Link Virtualization	57
5.1	Introduction	57
5.2	Test for Virtualization	57
5.3	Survey Overview	59
5.4	Non-Recursive Techniques	59
5.4.1	Frequency-based isolation	61
5.4.2	Frequency- and Time-based isolation	61
5.4.3	Space-based isolation	61

5.5	Recursive Techniques	62
5.6	Spectrum Level Virtualization	63
5.6.1	Frequency-based Isolation	63
5.6.2	Frequency and Time based Isolation	65
5.6.3	Time and Space based Isolation	66
5.6.4	Discussion	66
5.7	Information Level Virtualization	68
5.7.1	Time-based Isolation using Switching	68
5.7.2	Time-based Isolation using Addressing	69
5.7.3	Space-based Isolation	71
5.7.4	Discussion	71
5.8	Comparison with Existing Surveys	72
5.8.1	Focus on Purpose / Technology	72
5.8.2	Lack of Specificity	73
5.9	Insights & Discussion	74
5.9.1	Use of Multiple Representation Levels	74
5.9.2	Exclusive Analysis of Either Isolation or Embedding	75
5.9.3	Lack of Focus on the Isolation Problem	76
5.9.4	Exclusive use of Many-to-one Embedding	76
5.9.5	Lack of Isolation at the Spectrum Level	77
5.10	Acting on the Insights	77
6	Spectrum Level Virtualization: The Isolation Problem Part 1	
	<i>Designing an implementation</i>	79
6.1	Introduction	79
6.2	Design Principles	80
6.3	Evaluating Existing Implementations	82
6.3.1	Overview of Spectrum Virtualization Layer (SVL) Architecture	83
6.3.2	Signal Reshaping Operations	85
6.3.3	Implementation	89
6.3.4	Evaluation of SVL as Virtualization Research Platform	90
6.4	Evaluating Platforms for Our Implementation	91
6.4.1	GNU Radio	92
6.4.2	Iris	92
6.4.3	LuaRadio	92
6.5	Designing Our Implementation	93
6.5.1	Design Process	94
6.5.2	Overview of the Resulting Design	94
6.5.3	Detailed Design	96

6.6	Proof-of-Concept Simulations	105
6.6.1	Simulation 1: Understanding Spectrum Mapping	105
6.6.2	Simulation 2: Proving FFT Independence	110
6.6.3	Simulation 3: Testing Recursion	111
6.6.4	Simulation 4: Verifying Many-to-many Operation	114
6.6.5	Conclusions	115
6.7	Summary	116
7	Spectrum Level Virtualization: The Isolation Problem Part 2	
	<i>Performing isolation over-the-air</i>	117
7.1	Introduction	117
7.2	Introducing the CONNECT Testbed	118
7.2.1	Involvement in Testbed Creation	120
7.3	Experimental Process	120
7.4	Experiment 1: Establishing Baseline Performance	122
7.4.1	1a) Contiguous Spectrum Map	122
7.4.2	1b) Contiguous Spectrum Map with Guard Bands	125
7.4.3	1c) Non-Contiguous Spectrum Map	127
7.4.4	Observations	128
7.5	Frame Synchronization	128
7.5.1	Developing a Triggered Demultiplex Block	130
7.5.2	Testing Synchronization on the Testbed	131
7.5.3	Removing the Artefacts	134
7.6	Experiment 2: Adding Frame Synchronization to MySVL	134
7.6.1	Observations	137
7.7	Carrier Frequency Offset Estimation and Correction	139
7.7.1	Testing CFO Estimation and Correction on the Testbed	140
7.8	Experiment 3: MySVL with CFO Estimation and Correction	142
7.8.1	Observations	142
7.9	Potential Applications	144
7.9.1	Multiple Virtual Links per Real Link (many-to-one)	144
7.9.2	Combining Real Links into a Virtual Link (one-to-many)	145
7.9.3	Additional Applications	146
7.9.4	Flexible Spectrum Use and Dynamic Spectrum Access	146
7.9.5	Physical Layer Security	150
7.10	Summary	152
8	Spectrum Level Virtualization: The Embedding Problem	155
8.1	Introduction	155

8.2	The Embedding Problem	155
8.3	System Model	158
8.3.1	Dynamic Embedding	159
8.3.2	Problem Formulation	160
8.4	Dynamic Embedding Algorithms	162
8.4.1	Static Karnaugh-map Algorithm	162
8.4.2	Dynamic Karnaugh-map Algorithm	163
8.4.3	Greedy Dynamic Algorithm	163
8.5	Performance Evaluation	165
8.5.1	Evaluation Metrics	165
8.5.2	Simulation Setup	165
8.5.3	Results and Discussion	166
8.5.4	Conclusions	170
8.6	Application of Dynamic Embedding: Public Safety	170
8.6.1	Introduction to PPDR networks	171
8.6.2	System Overview	172
8.6.3	Investigated Scenario	174
8.6.4	Performance Results	177
8.6.5	Conclusions	180
8.7	Summary	180
9	Conclusions	181
9.1	Introduction	181
9.2	Summary of the Thesis	181
9.3	Lessons Learned	183
9.3.1	Structured Analysis	183
9.3.2	Use of Multiple Representation Levels	184
9.3.3	Focus on WLV at the Spectrum Level	184
9.3.4	Separating the Isolation and Embedding Problems	185
9.3.5	Flexible and Generic Implementation	185
9.4	Future Work	186
9.4.1	Comparing Virtualization at Different Representation Levels	186
9.4.2	Implementing Embedding with MySVL	187
9.4.3	Wireless Node and Network Virtualization	188
	List of Acronyms	189
	Bibliography	193

List of Figures

2.1	The physical domain represented in the abstract domain	15
2.2	Representation and instantiation; abstract and physical evolution	17
2.3	Levels of representation	19
2.4	Design choice in levels of representation	20
2.5	Advantage of multiple levels of representation	21
2.6	Hardware and software from a representation perspective	21
3.1	Virtualization relies on representation	26
3.2	Virtualization mapping types	28
3.2	Recursive virtualization	31
3.3	Resource isolation	32
3.4	Isolation example	33
3.5	Resource embedding	34
3.6	Virtualization theory example: Virtual memory	36
4.1	Functions of the compute cycle	40
4.2	Process virtualization: time-sharing	41
4.3	Wired link virtualization and network interface card virtualization	44
4.4	Wireless link virtualization and RF front-end virtualization	46
4.5	Wired network virtualization	48
4.6	Software defined networking	50
4.7	Software defined networking as prerequisite to virtualization	50
4.8	Network function virtualization	52
4.9	Wireless link and wireless network virtualization	53
4.10	Software defined radio as prerequisite to virtualization	54
5.1	Summary of the aspects of wireless link virtualization	75
6.1	The isolation problem for wireless link virtualization	80

6.2	The Spectrum Virtualization Layer (SVL) sits under the physical layer of the OSI model.	83
6.3	SVL maps virtual spectrum bands to real spectrum bands.	84
6.4	Full architecture of SVL.	85
6.5	Components of the signal reshaper.	86
6.6	Signal decomposition and recomposition in the frequency-domain.	87
6.7	Overview of the The Many-to-many Spectrum Virtualization Layer (MySVL)	95
6.8	Class diagram for MySVL	97
6.9	Example of an FFT parameters text file	99
6.10	Spectrum map text file	101
6.11	GNU Radio Companion (GRC) flowgraph for Simulation 1	105
6.12	Simulation 1a: contiguous virtual and real spectrum	107
6.13	Simulation 1b: non-contiguous real spectrum	108
6.14	Simulation 1c: non-contiguous real spectrum with a null signal	109
6.15	Simulation 1d: random spectrum assignment	110
6.16	Simulation 2: spectrum mapping with flexible FFT sizes	111
6.17	GRC Flowgraph for Simulation 3	112
6.18	Simulation 3: recursive spectrum mapping	113
6.19	GRC Flowgraph for Simulation 4	114
6.20	Simulation 4: many-to-many spectrum mapping	115
7.1	The CONNECT testbed	118
7.2	The interface of the reservation system	119
7.3	Private network with virtual nodes for running simulations	120
7.4	Transmitter flowgraph and signals for Experiment 1a	123
7.5	Receiver flowgraph and signals for Experiment 1a	124
7.6	Transmitter flowgraph and signals for Experiment 1b	125
7.7	Transmitter flowgraph and signals for Experiment 1b	126
7.8	Experiment 1c: non-contiguous real spectrum	127
7.9	Frame synchronization preamble detection	129
7.10	Transmitter flowgraph and signals for testing frame synchronization . . .	131
7.11	Receiver flowgraph and signals for testing frame synchronization	132
7.12	The flowgraphs for the synchronization test with padding.	133
7.13	Frame synchronization with padding	134
7.14	Experiment 2: transmitter flowgraph and signals	135
7.15	Experiment 2: receiver flowgraph and signals	136
7.16	The effect of frame lengths on frame error rates and overhead	138
7.17	Examining output of MySVL to observe signal distortion	139

7.18	Preamble used for carrier frequency offset estimation	140
7.19	Flowgraph and plots for testing CFO estimation and correction	141
7.20	Receiver flowgraph for MySVL with carrier frequency offset correction . .	143
7.21	Example of dynamic spectrum access using MySVL	147
7.22	Unfiltered and filtered real spectrum for the Secondary User (SU)	148
7.23	SU transmitter flowgraph	148
7.24	Real and virtual spectrum at the SU receiver.	149
7.25	Demonstration using MySVL for physical layer security	152
8.1	The embedding problem for wireless link virtualization	156
8.2	Example showing the advantages of dynamic embedding	159
8.3	Comparison of static vs dynamic embedding	166
8.4	Influence of larger request sizes on the embedding algorithms	167
8.5	Detailed rejection rates for small and large request sizes	168
8.6	Comparison of embedding algorithms to the optimal embedding	169
8.7	Virtual resource requests per timeslot	176
8.8	Rejection rates per traffic type and operator	178
8.9	Real resources occupied by traffic type and operator	179

List of Tables

2.1	Definition of terms	13
3.1	Virtualization theory example: Virtual memory	37
5.1	Wireless resource allocation techniques	60
5.2	Wireless link virtualization techniques: Isolation	64
5.3	Wireless link virtualization techniques: Embedding	67

1 Introduction

1.1 Introduction

This thesis is concerned with the virtualization of wireless links. Initially, the sole aim of this thesis was to investigate the virtualization of wireless networks. However, soon after first investigating this topic it became apparent that the term ‘virtualization’ is not used in a precise or meaningful manner in current technical work, and that the largest obstacle to wireless network virtualization is the virtualization of wireless links. Therefore, although investigating wireless link virtualization is the primary goal of the thesis, a secondary objective of the thesis is to clarify virtualization in a general context. This opening chapter of the thesis demonstrates the lack of meaning surrounding the term ‘virtualization’ in more detail, before laying out the objectives and contributions of this thesis.

1.2 Virtualization of Communication Networks

The term ‘virtualization’ features in abundance in any discussion on future communication networks as virtualization is envisaged as part of the long-term vision for communication networks, whether from an academic, industrial, or regulatory point of view.

In the academic world, a significant body of work exists on the virtualization of networks and network resources. A current search on IEEE Xplore for the term “Network Virtualization” returns close to 5000 academic works. Similarly, a significant number of projects exist which focus on network virtualization, among which are GENI [1], PlanetLab [2], 4WARD [3], SAVI, AKARI, and many others [4, 5]. The following examples illustrate some of the views on virtualization in the communication research community:

Network virtualization is one of the most promising technologies for the future Internet. [6]

Virtualization of wired networks and end computing systems has become one of the leading trends in networked ICT systems. In contrast relatively little virtualization has occurred in infrastructure based wireless networks, but the idea of virtualizing wireless access is gaining attention. [7]

Virtualization is widely accepted to be the basis for enabling a flexible Internet architecture for the future that would accommodate multiple architectures and allow disruptive innovations and technologies to be easily incorporated into the core architectures. [5]

In addition to academia, many companies are now actively involved in network virtualization research, and especially in Network Function Virtualization (NFV) research. A recent white paper published collectively by major telecom providers (such as AT&T, Deutsche Telekom, BT, China Mobile, etc.) states that:

Network Function Virtualization aims to transform the way that network operators architect networks ... We believe Network Function Virtualization is applicable to any data plane, packet processing and control plane function in fixed and mobile network infrastructures. [8]

Another report commissioned by Cisco states that “network virtualization represents a new IT paradigm, which challenges existing assumptions and IT deployment models ... In particular a new generation of network virtualization technologies is available which are becoming the new building blocks of network design.” [9]

Similarly, regulatory and standards bodies are also implementing new standards which include virtualization. The 3rd Generation Partnership Program (3GPP) reports that “it is expected that future implementation options for the entities of the 3GPP system include various technologies for virtualization.” [10]

1.3 The Many Meanings of Virtual

The proliferation of the term ‘virtualization’ throughout the above examples signifies that virtualization is a concept worthy of focus. However, despite the ubiquitous use of this term, its exact meaning is often not clear. The root of this problem is that the term ‘virtual’ is ambiguous and can refer to several different concepts, leading to disagreement and confusion. As noted by Russell in [11], the word ‘virtual’ has become so widespread as to render its use almost meaningless.

In vernacular usage, one of the most common meanings of virtual is “that may be so called for practical purposes, although not according to strict definition; very near, almost absolute” [Oxford English Dictionary (OED)]. For example, the phrase ‘virtually finished’ means that something is almost complete, but not quite complete. Another example is a phrase such as ‘a virtual prisoner’, meaning almost a prisoner. We term this use of the word ‘virtual’ as meaning one (*i*).

A second meaning (*ii*) is “that which is so in essence, potentiality, or effect, although not in form or actuality” [OED]. In this form, ‘virtual’ takes on the meaning of supposed, imagined, or simulated, and leads to phrases such as ‘virtual space’, ‘virtual team’, ‘virtual affair’, and many more.

Stemming from meaning two, a third meaning (*iii*) of the word ‘virtual’ is that of “not physically present as such but made by software to appear to be so from the point of view of a program or user” [OED]. This meaning has led to the word ‘virtual’ almost becoming synonymous with the word ‘digital’ in contemporary usage [11]. In fact, ‘virtual’ is now often used as a proper noun – ‘the virtual’ – as a world of “graphical objects and animated personae which populate fictional, ritual and digital domains as representatives of actual persons and things” [12].

Following from the above meaning, the virtual is often contrasted with ‘the real’ (meaning *iv*), and thought of as fake, fictitious, non-existing, or imaginary, whereas ‘real’ signifies concrete, tangible, and reliable [13], [12]. However, the philosophical meaning (*v*) of the term ‘virtual’ defines virtual as real (i.e. existing) but ideal rather than actual [14]. The virtual is a ‘real idealization’ such as a memory, a dream, or a company brand. In short, the philosophical use of virtual refers to *intangible* objects, which although not having physical presence, nonetheless exist [12]. In fact, the issue of what is real and what is not has been a long-standing philosophical debate, creating entire fields of philosophy known as ontology (studying what exists) and epistemology (the study of certainty about what exists).

Although such debate is outside the scope of this thesis, it is clear that there are many meanings of ‘virtual’, which are often incompatible and contradictory. One such example is the phrase ‘virtual reality’, which knowing the philosophical meaning, appears to be an oxymoron [15].

1.4 Technical Use of Virtualization

When the terms ‘virtual’, and ‘virtualization’ are used in a technical context, for example regarding the virtualization of computing resources and communication networks, the confusion and ambiguity surrounding these terms becomes even more pronounced. Some examples of the vague and sometimes contradictory definitions which exist in the relevant literature are provided below.

Forde et al. [16] describe virtualization as:

the creation of a virtual version of something, rather than the actual thing itself. The distinction between real and virtual in terms of computational entities is not very satisfactory. But for the purposes of describing the concept of a virtualized cellular network infrastructure let us think of virtual components as being those components that are brought into a useful state by the configuration, assembly and programming of generic components that would not operate without our intervention.

This definition starts with the use of meaning *ii* for the word virtual, but then moves to meaning *iv* when it distinguishes virtual as opposed to real. It acknowledges that this distinction is not clear, but the definition of virtual components as brought into a useful state is very ambiguous.

The definition of virtual as useful is reflected in the definition provided by Sahoo et al. [17], which defines virtualization in terms of its purpose:

Virtualization is a term that refers to the abstraction of computer resources. The purpose of the virtual computing environment is to improve resource utilization by providing a unified integrated operating platform for users and applications based on aggregation of heterogeneous and autonomous resources.

However, the above definition also refers to virtualization as the abstraction of resources. This phrase is used time and time again in the literature on computer and network virtualization, but it is fundamentally not clear whether the abstraction refers to the abstract (i.e. the virtual as opposite of the real, meaning *iv*), or whether the abstraction refers to the process of abstraction, which means to simplify, to take something away.

Similarly, the definition of virtualization in terms of purpose and the use of the term ‘abstraction’ can be found in the definition provided in Wen et al. [18]:

Wireless network virtualization is the abstraction and sharing of wireless hardware equipments by multiple virtual owners, referred to as tenants. The virtual instance of a set of shared resources is called a slice. Each tenant is given the illusion of ownership over its slice defined by a service-level agreement. However, the degree of sharing can vary from high-level mobile network management to low-level spectrum and radio frequency frontend slicing.

and again in the definition by Liang et al. [19, 20]:

We consider wireless network virtualization as the technologies in which physical wireless network infrastructure resources and physical radio resources can be abstracted and sliced into virtual wireless network resources holding certain corresponding functionalities, and shared by multiple parties through isolating each other.

In these definitions, several interesting points appear. Firstly, in both definitions virtualization is defined exclusively in terms of sharing (or ‘slicing’) resources between multiple users, without considering the possibility of combining or aggregating resources. Defining virtualization in such a manner introduces an artificial restriction and limits our understanding of virtualization. Secondly, again the concept of functionality or purpose is implied, indirectly by Wen et al., and directly by Liang et al. The final point to highlight, is the last sentence of the definition given by Wen et al. It brings up the notion of multiple levels.

The idea of multiple levels (whether of virtualization, or of abstraction is not clear) is at the heart of the confusion surrounding virtualization. It implies the use of abstraction again, but whereas previously abstraction might have been used to mean an abstract entity (i.e. the fourth meaning of virtual), now abstraction is used to mean the process – that is high-levels of abstraction having higher complexity than low-levels of abstraction.

There are several other definitions which avoid the use of the term abstraction, such as the definition is offered in Sachs et al. [21]. It states:

The process of sharing and allocating resources belonging to a physical radio link (i.e. a radio resource) we refer to as radio virtualization; the resources that are used by a particular virtual network node we refer to as virtual radio (i.e. a virtual link). The virtual instantiation of node functionality running on a physical network node we call virtual node. The coordination and management of physical resource allocation among a multitude of virtual radio

nodes is performed by a resource allocation control function. A virtual network consists of a set of links and nodes; by virtual radio network we refer to a virtual network in an edge network of the Future Internet that comprises multiple interconnected virtual radio nodes.

Although this definition is somewhat easier to follow than previous definitions, it still defines virtualization in terms of purpose, and only considers the sharing or splitting case.

Finally, the term virtualization is increasingly used (especially by the NFV community) to refer to the process of “softwarization” – the process of moving functionality from hardware to software. For example the authors of [22] refer to network function virtualization as “transferring network functions from dedicated hardware appliances to software-based applications running on commercial off-the-shelf equipment”. This use of virtualization reflects the third meaning of ‘virtual’ being synonymous with ‘digital’.

All in all, it is not easy to get a clear understanding of wireless network virtualization from these definitions. Firstly, almost all definitions define virtualization in idealistic terms, according to what its purpose or function should be, rather than what the process is. Secondly, the use of the phrase “abstraction of resources” (which is even more common in very recent definitions) is confusing as it is not clear which meaning of abstraction is meant. It seems as if both meanings are used indistinguishably to refer to different aspects of virtualization. Thirdly, the majority of definitions are only focused on virtualization as a splitting or sharing of resources, without considering the possibility of combining resources.

1.5 Objectives and Research Questions

It is the lack of clarity surrounding virtualization in the context of wireless networks that is the starting point of the thesis.

Therefore this thesis is guided by two themes; firstly, understanding virtualization in a technical context and developing a precise and meaningful definition of virtualization, and secondly, using the insights provided by a thorough understanding of virtualization to perform meaningful research on wireless network virtualization.

1.5.1 Developing a Precise Understanding of Virtualization

The first aim of the thesis is to produce a clear, exact definition of virtualization as a process, rather than in terms of its supposed purpose(s). Not only is it desirable from an

academic point of view to have a clear definition, but it is also a practical necessity. Since virtualization is expected to be such a crucial part of future networks, it is necessary to know what is meant when using this term, to establish a common understanding. Moreover, it cannot be possible to do innovative research on virtualization when its meaning is not certain, and existing research gaps can not be described clearly.

Research Question 1: How do we produce a precise and meaningful definition of virtualization?

As will be shown in Chapter 2, to answer the above question, it will be necessary to first explain the concept of *representation* in detail, as it is a prerequisite to virtualization. Having gained an understanding of representation, we will develop a theory of virtualization in Chapter 3, before constructing a meaningful technical definition of virtualization.

1.5.2 Using the Precise Understanding of Virtualization

Armed with a rigorous definition and theoretical framework of virtualization, a natural question to ask is what purpose the definition serves. Therefore the second objective of the thesis is to use the definition to bring about relevant research in the area of wireless network virtualization.

Research Question 2: How can we use our precise theory of virtualization to do research on wireless network virtualization?

As a first step in answering this question, in Chapter 4 we will explore virtualization in the context of communication networks, guided by the theory of virtualization. From this exploration, it will become clear that the largest obstacle to wireless network virtualization is wireless link virtualization. Accordingly, we will survey the existing work on wireless link virtualization in Chapter 5 (although the literature often self-identifies as wireless *network* virtualization). As will be seen, this survey provides many new insights, but as will become evident, there are three insights which stand out most clearly, which have been overlooked by the research community.

Therefore, as a next step in answering the above research question, we act on these insights and investigate the corresponding research problems. The first insight concerns the lack of work on a part of virtualization that will become known as the isolation problem.

A second insight from the survey is that a form of virtualization called spectrum-level virtualization has received relatively little attention. Therefore bearing these two insights in mind, Chapters 6 and 7 will investigate the isolation problem for virtualization at the spectrum level, bearing in mind the definition of virtualization, which will lead to an implementation of wireless link virtualization. Additionally, we will describe and demonstrate several potential applications of spectrum-level wireless link virtualization.

The third insight from the survey is that the two sub-problems which make up virtualization, the isolation problem and the embedding problem as will be described in detail later, are often examined independently without consideration for the other problem. However, as will be shown, the embedding problem depends on the isolation problem, and as a result of the lack of coordination between the two problems, much work tackling the embedding problem cannot be used practically with the isolation problem. Therefore, in Chapter 8 we will investigate the embedding problem, but keeping in mind the constraints of the isolation problem. To make contributions in this already significant research field, we will propose dynamic embedding as a means of improving flexibility and efficiency.

Lastly, the concluding chapter of the thesis will discuss the benefits of the approach we will take throughout the thesis and the wide range of research possibilities that the thesis opens up. Overall lessons learned and additional work which we hope to complete in the future will also be described.

1.6 Key Contributions

While answering the above research questions, we will make the following contributions in two areas:

1.6.1 Developing a Precise Definition of Virtualization

- We will clarify the concepts of abstraction and representation, which are key to understanding virtualization, by drawing on Abstraction/Representation (AR) theory [23]. We will extend AR theory by considering multiple levels of representation in the abstract domain.
- We will provide a rigorous definition of hardware and software, which is surprisingly hard to find in the current literature.

- We will propose a formal method for describing virtualization, which we will call the theory of virtualization, and provide a precise definition of virtualization.

1.6.2 Using the Precise Definition of Virtualization

- We will develop a test for virtualization to distinguish virtualization techniques from resource allocation techniques, and will use this test to survey existing work on wireless network virtualization. Guided by the theory of virtualization, we will classify this work in a coherent and meaningful manner, and identify several research gaps in wireless network virtualization which have not yet been addressed.
- Addressing one research gap, we will design a many-to-many spectrum-level implementation for wireless link virtualization in a flexible and modular manner. Through simulation, we will show that our implementation achieves isolation between virtual link resources.
- We will perform spectrum level wireless link virtualization over-the-air on a test-bed, and will investigate measures needed to allow contiguous and non-contiguous spectrum virtualization. Having developed an implementation, we will demonstrate several potential applications of wireless link virtualization at the spectrum level.
- We will investigate the embedding problem for wireless link virtualization and propose dynamic embedding algorithms. We will compare these algorithms in terms of rejection rate and revenue and show the benefits of dynamicity.
- We will formulate the two-dimensional knapsack problem with priority constraints for static and dynamic embedding.
- We will propose another application of virtualization – a virtual public safety operator using a commercial network – and show the benefits of dynamic embedding at the spectrum level in this context.

1.7 Thesis Outline

The remainder of the thesis is organised as follows:

Chapter 2 – Prerequisites to Virtualization This chapter introduces Abstraction/Representation theory developed in by the authors of [23], and extends AR

theory with the concept of levels of representation. We show how AR theory is necessary to understand virtualization.

Chapter 3 – A Theory of Virtualization Building on AR Theory, virtualization theory is proposed in this chapter, and we arrive at a precise definition of virtualization. We provide a practical example, showing how virtualization theory applies to existing virtualization techniques.

Chapter 4 – Virtualization in Wireless Networks This chapter discusses the development of different virtualization technologies in networking, and how these technologies are needed for wireless network virtualization.

Chapter 5 – Survey of Wireless Link Virtualization Using the clarity provided by virtualization theory, this chapter surveys the existing literature on wireless link virtualization, and identifies open research questions.

Chapter 6 – Spectrum Level Virtualization: The Isolation Problem 1 As the survey shows a lack of focus on the isolation problem for spectrum-level virtualization, in this chapter we design and implement a many-to-many spectrum-level virtualization technique, and test our implementation through simulations.

Chapter 7 – Spectrum Level Virtualization: The Isolation Problem 2 Having verified that the implementation is working correctly, in this chapter we perform over-the-air experiments on the CONNECT testbed. We develop additional measures needed to do wireless link virtualization in real world conditions.

Chapter 8 – Spectrum Level Virtualization: The Embedding Problem This chapter examines the embedding problem, keeping in mind the isolation problem, and compares static and dynamic embedding for wireless link virtualization at the spectrum level. Additionally, we examine the case study of a public safety virtual operator sharing a network with a commercial virtual operator and show the benefits of virtualization and dynamic embedding.

Chapter 9 – Conclusion The last chapter summarizes the main findings, and discusses the advantages of the systematic approach taken in this thesis and the lessons we learned. Lastly, we describe a range of possible future research directions, which has been enabled by the approach taken in this thesis.

1.8 Publications

A number of works have been published during the course of this research. Publications relevant to this thesis are marked as ●, while other publications are marked as ○.

Journal Articles

- **J van de Belt**, H. Ahmadi, and L. E. Doyle, “Defining and surveying wireless link virtualization and wireless network virtualization,” *IEEE Communications Surveys and Tutorials*, vol. 19, no. 3, pp. 1603–1627, Third Quarter 2017.

Conference Papers

- P. Alvarez, C. Galiotto, **J. van de Belt**, D. Finn, H. Ahmadi, L. DaSilva., “Simulating dense small cell networks,” in *Wireless Communications and Networking Conference (WCNC), 2016 IEEE*, pp. 1–6.
- **J. van de Belt**, H. Ahmadi, L. E. Doyle and O. Sallent, “A prioritised traffic embedding mechanism enabling a public safety virtual operator,” in *Vehicular Technology Conference (VTC Fall), 2015 IEEE 82nd.*, pp. 1–5.
- J. Tallon, C. Blumm, A. Puschmann, F. Paisana, **J. van de Belt**, P. Di Francesco, N. Kaminski, H. Ahmadi “Coexistence through adaptive sensing and Markov chains,” in *Dynamic Spectrum Access Networks (DySPAN), 2015 IEEE International Symposium on.*, pp. 7–8.
- **J van de Belt**, H. Ahmadi, and L. E. Doyle, “A dynamic embedding algorithm for wireless network virtualization,” in *Vehicular Technology Conference (VTC Fall), 2014 IEEE 80th.*, pp. 1–6.
- **J. van de Belt**, P. D. Sutton, and L. E. Doyle, “Accelerating software radio: Iris on the Zynq SoC,” in *Very Large Scale Integration (VLSI-SoC), 2013 IFIP/IEEE 21st International Conference on.*, pp. 294–295.

2 Prerequisites to Virtualization

2.1 Introduction

Virtualization is often defined as the abstraction and sharing of resources. However, as discussed in Chapter 1, the use of the term ‘abstraction’ here is vague and meaningless, and it blurs the distinction between virtualization and softwarization. Therefore, we address the question of producing a precise and meaningful definition of virtualization.

There are a number of concepts, which are key to developing a precise definition of virtualization and in this chapter we carefully define these concepts, before developing a precise definition of virtualization in the following chapter. The first two of these are the concepts of *abstraction* and *representation*. Table 2.1 shows an explanation of these and other terms used in this thesis.

In this thesis, the term ‘abstraction’ means the act of ignoring or hiding details to consider general characteristics, rather than concrete realities. Thus abstraction manages the way in which systems interact, and the complexity of the interaction, by hiding details that are not relevant to the interaction. Increased abstraction allows systems to be used more easily for specific applications, but this comes at the cost of decreased flexibility and customization. Although abstraction is an important concept in computing, as it governs the interaction between humans and computers, it is not necessarily of importance to

Table 2.1: Definition of terms

Term	Explanation
Abstraction	The act of ignoring details to consider general characteristics
Abstract	Existing as a thought, idea, or concept, but without physical existence
Representation	The act of symbolising or portraying something in a particular way
Instantiation	Creating a concrete realization of a concept or idea

virtualization. However, it is important to note the difference between abstraction and the adjective ‘abstract’. The term ‘abstract’ refers to ideas and concepts that do not have physical existence.

The term ‘representation’ is used to mean the portrayal or symbolization of something in a particular way. The term ‘instantiation’ means the implementation or realization of a concept or idea. Representation and instantiation are very important to computing as they describe the relationship between abstract entities and the physical world. As will be shown later in this chapter, representation and instantiation are of great importance to virtualization, since virtualization can only be done in the abstract domain, while network resources exist in the physical domain.

We now turn to the recently developed Abstraction/Representation (AR) theory, to provide us with a formal framework of abstraction and representation [23], [24]. Because this theory is very new, we explain the most important aspects here, borrowed from [23], before we go on to extend the ideas for the purposes of this thesis.

2.2 Abstraction/Representation Theory ¹

Abstraction/Representation theory ² is concerned with the physical domain and the abstract domain (also known as the logical domain), and the relationship between these domains.

The physical domain, $\mathbf{P}$, is defined as consisting of all physical objects, $\mathbf{p} \in \mathbf{P}$. The abstract domain, M , consists of all abstract objects, $m \in M$. For instance, a *computer* is an object in the physical domain, which can be in different physical states, while a *computation* is a set of objects and relations in the abstract domain. Bold font is used to indicate an object in the physical domain; italic font for an object in the abstract domain.

2.2.1 Representation

A physical object can be represented in the abstract domain, through a representational relationship, R . For example, a physical on-off switch can be represented in the abstract domain by a binary digit, shown in Figure 2.1. The general representation relation between a physical object, $\mathbf{p}$, and an abstract object, $m_{\mathbf{p}}$, is through a directed map

¹Based on [23]

²AR Theory might be better named as Representation Theory, since it deals mostly with representation, but that name has already been taken.

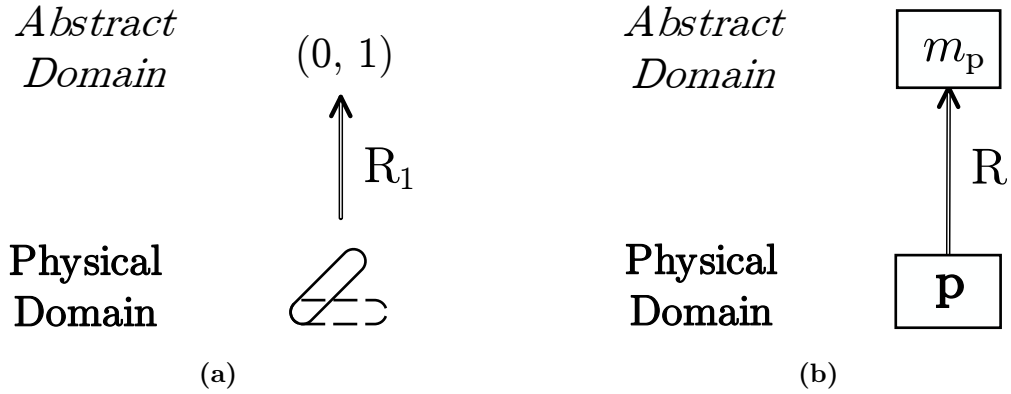


Figure 2.1: Objects in the physical domain can be *represented* using objects in the abstract domain, such as (a) a switch with two settings represented as a bit through R_1 , or (b) more generally as an object $\mathbf{p}$ represented as $m_{\mathbf{p}}$ through R .

$R : \mathbf{p} \rightarrow m_{\mathbf{p}}$. The abstract object, $m_{\mathbf{p}}$ is said to be an abstract *representation* of the physical object $\mathbf{p}$.

It is very important to note that the representation relation is *not* a mathematical function or a logical relation, but rather a modelling relation that bridges the divide between the physical and the abstract spaces. In other words, the representation relation is the relationship that allows humans to give abstract descriptions to physical objects, e.g. an atom is represented as a wavefunction, a physical ball represented as a circle, or a musical sound as a note. The reason why the representation relation cannot be a mathematical function is that since mathematics is completely in the abstract domain, and the representation relation operates between the physical and the abstract domains, the representation relation cannot be part of mathematics. For additional discussion and examples of why the representation relation cannot be a mathematical function, see the works [23, 24].

The representation relation is a *convention* which enables humans to describe physical objects without their presence, e.g writing (see [25]), or to derive abstract relations from physical objects, e.g. the measurement of one particular aspect of a physical object. In fact, measurement theory explains how certain attributes of objects are represented numerically, often as attributes which are observed to always be true [26]. The foundations of these representations rely on empirical measurements, meaning the physical domain in the context of AR theory. However, from time to time, some attributes are shown not to be completely true, such as when physics develops a new theory, for example when the theory of relativity showed that Newtonian physics did not always hold true, thus demonstrating that empirical measurements can be flawed and that the attributes assigned to physical objects are not always constant [26].

It is of vital importance to emphasise that the representation relation is not a mathematical function, but an (arbitrary) modelling relation.

2.2.2 Physical and Abstract Evolution

In the abstract domain, there can be an evolution or process, $C : m_{\mathbf{p}} \rightarrow m'_{\mathbf{p}}$, that changes an abstract object $m_{\mathbf{p}}$ to another abstract object $m'_{\mathbf{p}}$. Similarly in the physical domain, a corresponding evolution $\mathbf{H} : \mathbf{p} \rightarrow \mathbf{p}'$, changes the physical state $\mathbf{p}$ to physical state $\mathbf{p}'$. This physical state, $\mathbf{p}'$ can then be represented as $m_{\mathbf{p}'}$, through the representation relation, R . These concepts are shown in Figure 2.2 (a).

The two abstract objects, $m_{\mathbf{p}}$ and $m'_{\mathbf{p}}$, lead us to a key concept in AR theory. If $|m_{\mathbf{p}'} - m'_{\mathbf{p}}| \leq \varepsilon$, for some error ε and norm $||$, then we can say that the abstract evolution, C , and the corresponding physical evolution, $\mathbf{H}$, *commute*. Under the above condition, the two representation relationships, $R: \mathbf{p} \rightarrow m_{\mathbf{p}}$ and $R: \mathbf{p}' \rightarrow m_{\mathbf{p}'}$, and the pair of abstract and physical evolutions $C : m_{\mathbf{p}} \rightarrow m'_{\mathbf{p}}$ and $\mathbf{H} : \mathbf{p} \rightarrow \mathbf{p}'$, can be said to form a *commuting diagram*. When a set of abstract and physical objects form a commuting diagram using the representation, R , then $m_{\mathbf{p}}$ is a *faithful abstract representation* of physical system $\mathbf{p}$ for the evolutions $C(m_{\mathbf{p}})$ and $\mathbf{H}(\mathbf{p})$. This means we can be confident that the evolution C in the abstract domain corresponds to the evolution $\mathbf{H}$ in the physical domain.

The implication of commuting diagrams is that the abstract representation of the final state of a physical object (i.e. $m'_{\mathbf{p}}/m_{\mathbf{p}'}$) can be found either by following the physical evolution and then representing the output abstractly, or by theoretically evolving the representation of the physical state. As an example, consider a commuting diagram in which physical voltages are represented by binary numbers, shown in Figure 2.2 (b). Assume that we want to perform an abstract evolution (binary addition). Then this evolution can be performed either in the abstract domain using logic gates, or through physical manipulation of voltages in the physical domain and representing the result abstractly.

2.2.3 Instantiation

The instantiation relationship, $\tilde{R}$, can be thought of as the inverse to the representation relation. Just as a physical object can be represented in the abstract domain by the representation relation, the instantiation relation, $\tilde{R}: m_{\mathbf{p}} \rightarrow \mathbf{p}$, instantiates an abstract object in the physical domain.

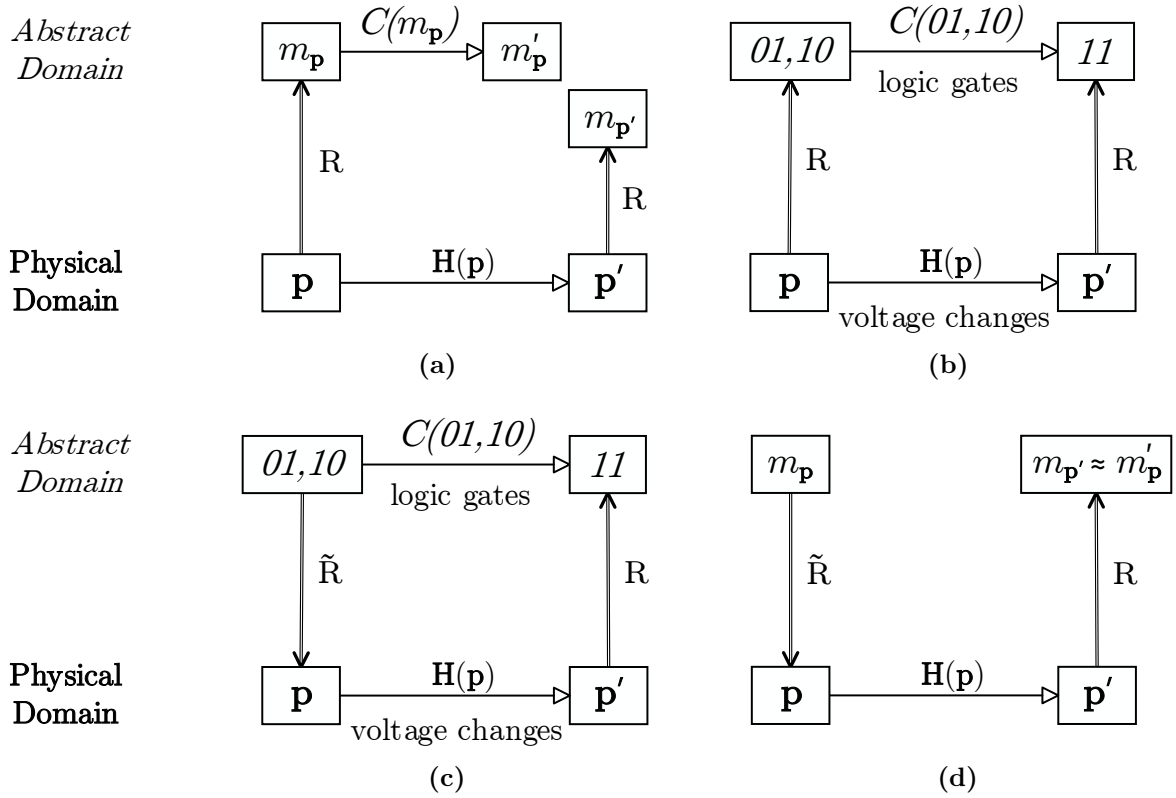


Figure 2.2: (a) A physical system p and its representation m_p can undergo abstract evolution, $C(m_p)$, or physical evolution, $H(p)$. (b) A commuting diagram for a binary addition, showing that an abstract evolution through logic gates, C , is commutative with a physical evolution in voltage states, H . (c) A computer can evaluate an abstract evolution by instantiating the abstract object in the physical domain, performing the physical evolution, and representing it as an abstract object. (d) The most interesting use of a computer is when the abstract evolution is unknown.

However, unlike representation, the instantiation relation can only exist under specific conditions, as there are many abstract objects that have no physical instantiation. It is necessary for a commuting diagram to exist for a given representation, before any attempts can be made to find the inverse instantiation relation (see [23]). Finding an instantiation relation is not straightforward and can be thought of as finding a physical system, that when represented abstractly gives the abstract object that we desire. This often requires trial and error, if such an instantiation is even possible.

Figure 2.2 (c) shows the binary addition example, but this time the voltages instantiate binary numbers. The instantiation relation, $\tilde{R}$, can be used to change the physical state to p , so that it instantiates the numbers we wish to add. In the abstract domain, the abstract mapping (i.e. mathematical and logical operations) C , performs the addition to arrive at the result. Meanwhile, the physical mapping, H , manipulates the voltages to produce the physical result. Using the representation relation, R , the abstract representation of p' is found. If we have confidence that the representation is a

faithful abstract representation and also that the instantiation relation is correct, then the outcome of the abstract and the physical evolutions should be the same.

2.2.4 The Compute Cycle

The previous example describes a computer performing a parallel operation in the abstract and physical domains. However, the most interesting use of a computer is when the abstract mapping C is unknown and we can use the computer to solve an abstract problem, shown in Figure 2.2 (d). Provided that we are confident in the capabilities of the computer, we can use the computer to find the solution.

The full compute cycle is as follows:

$$m_{\mathbf{p}} \xrightarrow{\tilde{R}} \mathbf{p} \xrightarrow{\mathbf{H}} \mathbf{p}' \xrightarrow{R} [m_{\mathbf{p}'} \approx m'_{\mathbf{p}}]$$

Thus representation and instantiation enable physical computing resources to implement abstract objects and operations, which can be called abstract resources.

2.3 Levels of Representation

We extend AR theory presented in [23] by focusing on the abstract domain and examining abstract objects in more detail. Most importantly, we observe that objects in the abstract domain, which represent physical objects, can in turn be represented by *different* abstract objects. The same is true for instantiation; objects in the abstract domain, which are instantiated in the physical domain, can instantiate *different* abstract objects. For the purpose of brevity, from here on we only discuss representation and imply that the same is true for instantiation.

In essence there are many different representations that can be used. We use the term ‘levels of representation’ to capture this idea and this is depicted in Figure 2.3. In this figure there are n different levels of representation in the abstract domain. We number these representation levels (RLs) from 1 to n , but the numbering is arbitrary and is only an identifier.

It is important to note that the representation used is an arbitrary design choice, and that it is possible to represent the physical domain at any representation level. The

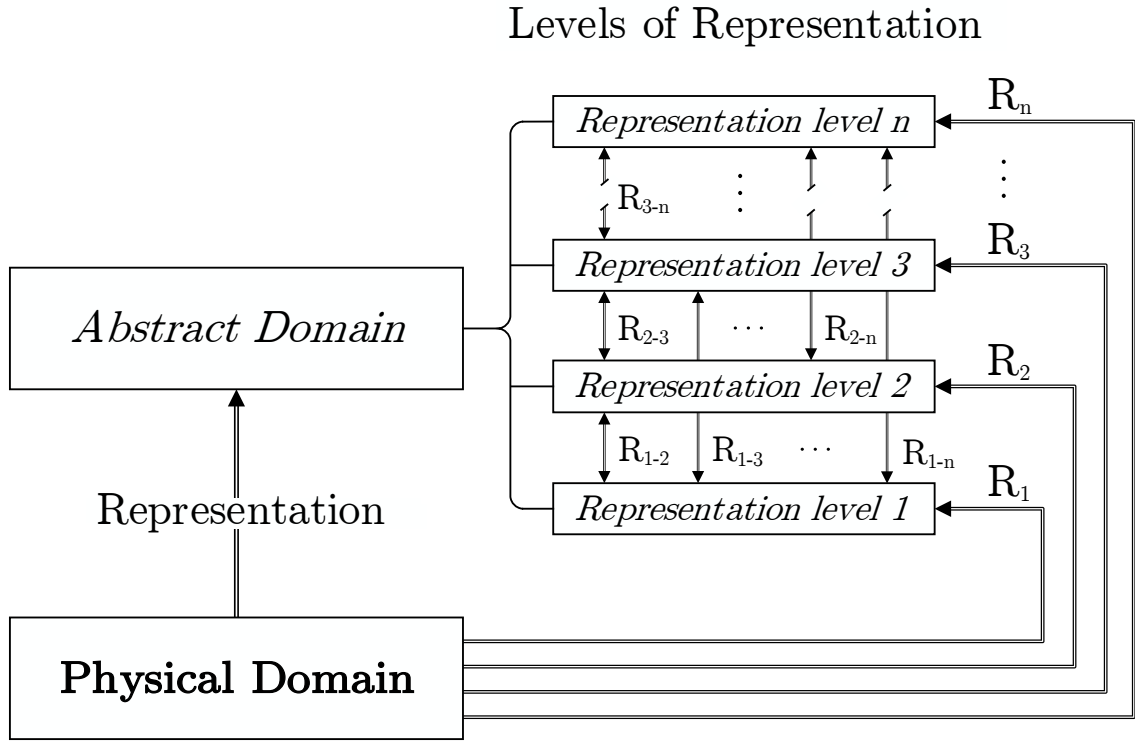


Figure 2.3: The abstract domain can be divided into many levels of representation, since abstract objects can represent other abstract objects. In this case there are n representation levels, ordered arbitrarily, but there can be an infinite number of levels. Each representation level can represent the physical domain directly, through unidirectional representations R_1 , R_2 , etc., or can represent another representation level, through the bidirectional representations R_{1-2} , R_{2-3} , etc. These representations are bidirectional to show that any representation level can represent another level. This image shows the theoretical representation relations, which does not mean that these relations will all exist in practice.

unidirectional arrows, R_1 , R_2 , etc., show the one-way representation relation from the physical to the abstract domain as seen previously.

Similarly, any representation level can represent another level, shown in the figure through the two-way representation relations R_{1-2} , R_{2-3} , etc.

An important point to observe is that there can be an infinite number of representation levels, as any abstract object can represent another abstract object. However in practice, not every representation level m will have a physical representation relation (i.e. the one-way representation R_m).

A practical example of the concept of levels of representation is shown in Figure 2.4 (a). In this example, a physical object can be represented by the abstract letter ‘k’ directly, and one representation level exists. However, Figure 2.4 (b) has two representation levels. In this case the letter ‘k’ represents a binary number, which in turn represents the physical domain.

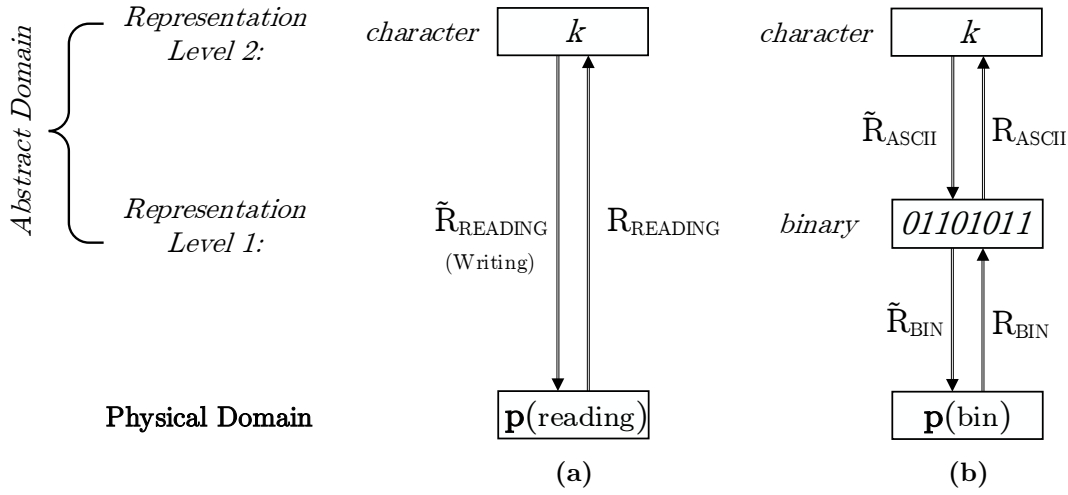


Figure 2.4: The number of representation levels is a design choice. For example, the character ‘k’ can (a) directly represent the physical domain, which we commonly know as reading, or (b) represent a binary number, which represents the physical domain. We can speak of one representation level in the first case, the character level, and two representation levels in the second case, the character level and the binary level.

2.4 Choosing a Hierarchy of Representation Levels

In principle, there can be an infinite number of representation levels in the abstract domain, with an arbitrary ordering of representation levels. However, in practice it is more useful to computing (and other abstract domain applications) if an ordered hierarchy of representation levels exists. Then it is possible to think of lower representation levels that are more concrete, and higher representation levels that are more abstract. When levels of representation are used in this manner, then we can number the representation levels in order of increasing abstraction.

One advantage of having an ordered hierarchy of representation levels is that existing physical instantiations for abstract objects can be reused, since it is not an easy task to design a physical instantiation of an abstract system [23]. Rather than finding a physical instantiation for an abstract system, an instantiation can be found in terms of an abstract system which already has a physical instantiation ³.

Figure 2.5 shows the advantage of multiple representation levels. In this case, the existing physical instantiation of binary numbers can be used to instantiate additional abstract objects. Many types of data such as integers, characters, volume levels, image brightness, and instructions, can be instantiated in computing using bits [27]. Thus, using multiple levels of representation provides flexibility and easier instantiation. In

³Which is not an easy task either, but easier than physical instantiation.

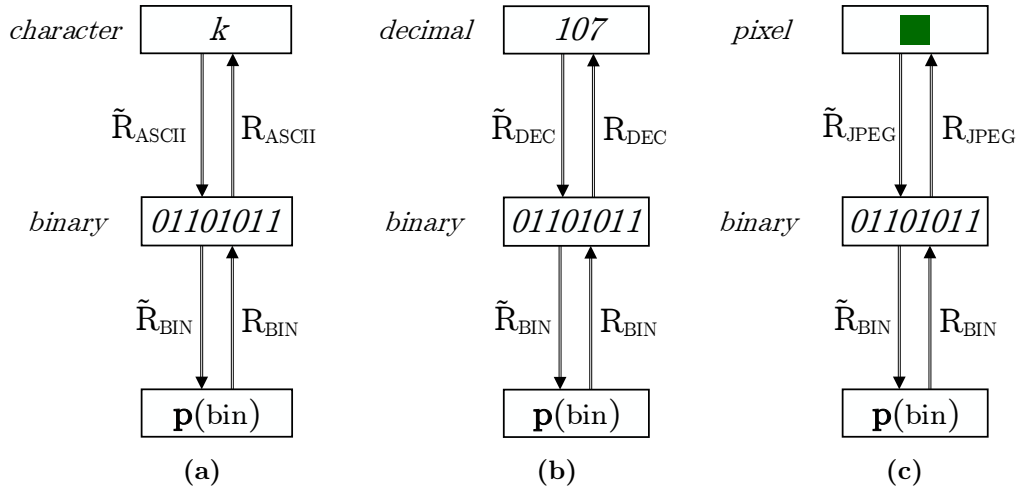


Figure 2.5: Multiple levels of representation enable a single physical instantiation to be reused for several different abstract objects. In this example, depending on the representation/instantiation, a binary number could instantiate (a) a character, (b) a decimal number, or (c) a pixel in an image.

this example, the binary representation level can be considered more concrete, and the character/decimal/pixel representation level can be considered more abstract.

2.5 Hardware and Software through Representation

The concept of levels of representation fits in very well with the idea of “hardware” and “software”. We define hardware as a compute cycle in which the representation/instantiation occurs between the physical and the abstract domain. Software, in contrast,

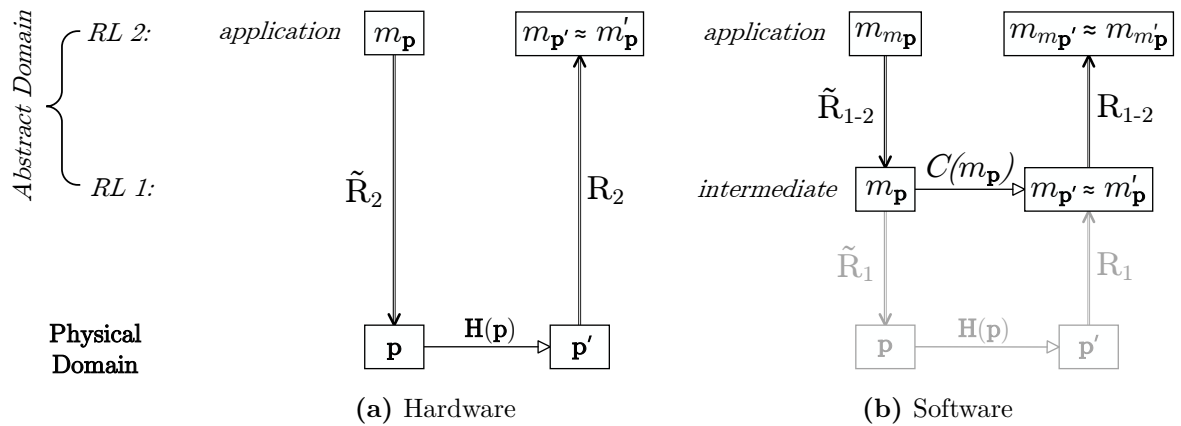


Figure 2.6: (a) Model of computer hardware and software based on AR theory. Computer hardware (a) can be thought of as performing the compute cycle in the physical and abstract domains, whereas software (b) performs the compute cycle in the abstract domain.

is defined as a compute cycle in which the representation/instantiation is completely in the abstract domain. A visual description of this distinction is shown in Figure 2.6.

We observe that the concept of representing the physical domain at any representation level (for example in Figure 2.4) is consistent with the principle of equivalence of hardware and software, which states:

Hardware and software are logically equivalent. Any operation performed by software can also be built directly into the hardware and any instruction executed by the hardware can also be simulated in software. [28]

The distinction between hardware and software will be useful later in this thesis (for example in Chapter 4).

2.6 How Representation applies to Virtualization

We have examined abstraction and representation, firstly to clarify the distinction between these concepts, and secondly because representation plays a role in virtualization in the following ways:

1. Virtualization must always be done in the abstract domain. The reason for this is that physical resources cannot be shared or combined without modifying their physical properties in some way. For example, it is not possible to split one processor physically to create multiple virtual processors, however it is possible to split a representation of a processor in the abstract domain. Virtualization allows abstract resources to be reorganised in a manner that is not limited by the underlying physical resources.
2. Since there can be many different representation levels in the abstract domain, virtualization can be performed at any of these representation levels and virtualization happens *within* a representation level. However, virtualization does not happen across a representation level - that is, the act of virtualization does not change abstract resources from one representation level to another representation level. As we will see later, it is important to know the representation level when virtualizing abstract resources.

2.7 Summary

We began this chapter by examining AR theory and explaining its most important aspects here, borrowed from [23]. Next, we extended AR theory with the idea of multiple levels of representation, which enables abstraction (managing the complexity of interaction between physical and abstract domains). The idea of levels of representation also allowed us to define hardware and software in a rigorous manner. Finally, we showed how representation applies to virtualization, which will be very important to formulate a theory of virtualization in the next chapter.

3 A Theory of Virtualization

3.1 Introduction

Having gained a better understanding of abstraction and representation, in this chapter we aim to address the first goal of the thesis: to produce a precise definition of virtualization as a process. We identify universal concepts of virtualization, and propose a theory of virtualization. Although some of the concepts have been mentioned before in the literature, to the best of our knowledge this is the first time that these concepts are brought together into a unified theory. From the theory, we produce a clear and precise definition of virtualization.

3.2 Virtualization as Resource Mapping

As shown in Chapter 2, resources in the abstract domain are representations of the physical domain. To be consistent with the language that is typically used in virtualization, we refer to these resources as *Real Resources (RRs)*. Although the term “real” is used, these resources are in the abstract domain, and are not physical resources. Also note that the act of representation is not virtualization, rather representation is a prerequisite for virtualization.

Real resources in the abstract domain can subsequently be virtualized. Virtualization is always performed in the abstract domain and at a specific representation level. Virtualization is a resource mapping which can alter the quantity of resources in some dimension(s). The resources after the virtualization process has occurred are referred to as *Virtual Resources (VRs)*, as shown in Figure 3.1. Virtual resources appear to be the same type of resources as real resources, but can be altered in quantity in some way. This altering allows abstract resources to be used in a flexible manner, not limited by

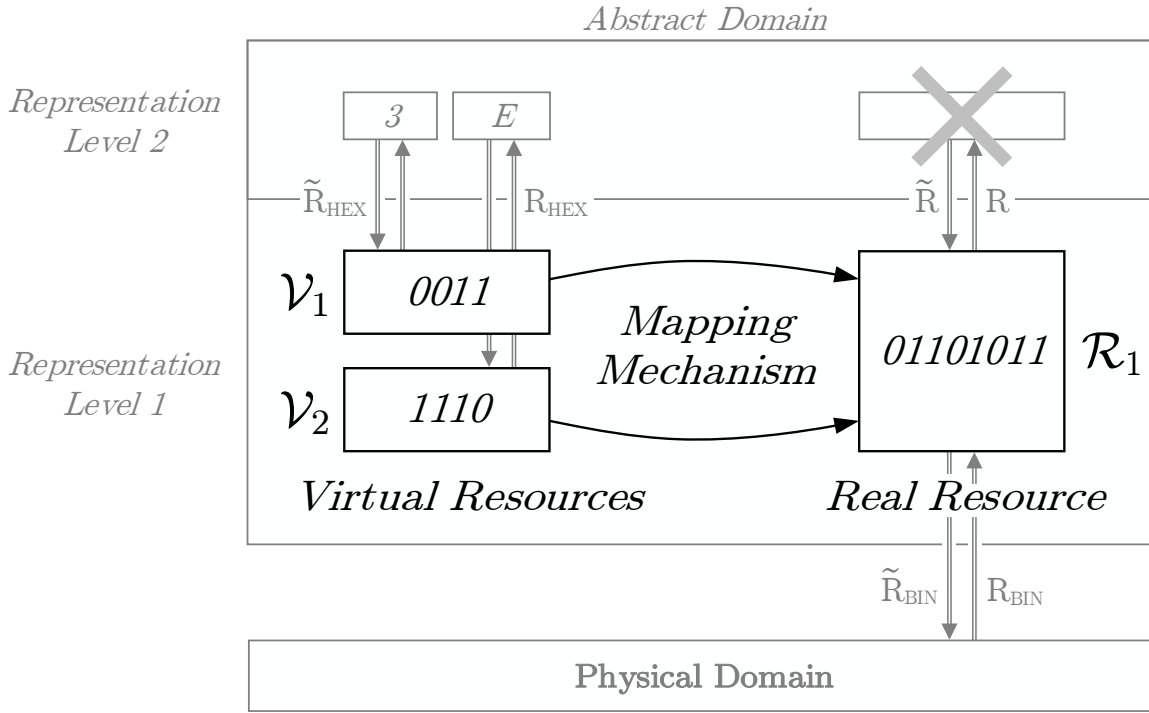


Figure 3.1: Representation allows physical resources to implement abstract resources, while virtualization allows abstract resources to be owned and used at tailor-made quantities. In this example the 8-bit abstract resource $\mathcal{R}_1$ is not an efficient resource to instantiate a hexadecimal number, as hexadecimal numbers only require 4-bits. Mapping two 4-bit virtual resources, $\mathcal{V}_1$, and $\mathcal{V}_2$, to $\mathcal{R}_1$ using a mapping mechanism is a more efficient use of this resource. The virtual resources $\mathcal{V}_1$, and $\mathcal{V}_2$ can then be used to instantiate hexadecimal numbers using the instantiation relation $\tilde{R}_{\text{HEX}}$. The real resource cannot be used to instantiate any higher representation level, shown by the cross in the figure.

the underlying representation of the physical domain. However, when virtual resources are mapped to real resources, the real resources cannot be used for any other purpose.

Virtual resources are used as if they were real resources, and it should not be possible to perceive any difference between virtual resources and real resources. In this thesis we consider that virtual resources are offered to one or multiple users. The term ‘users’ in this case refers to independent agents, that make decisions on resource use independently. Virtual resources can be offered to different users, and each user has the illusion of full ownership of the resources, meaning that virtual resources can be used for differing purposes.

In Figure 3.1, the real resource, $\mathcal{R}_1$, is an 8-bit number. This 8-bit number could be used to instantiate a hexadecimal number, or any other instantiation that uses 8 or less bits. However, a hexadecimal number only requires 4 bits to be instantiated, and using $\mathcal{R}_1$ would not be an efficient use of resources. Mapping two 4-bit virtual objects, $\mathcal{V}_1$, and $\mathcal{V}_2$ to $\mathcal{R}_1$, would be more efficient, and allows two hexadecimal (or any other 4-bit instantiation) to be instantiated, using the same resource, $\mathcal{R}_1$.

Virtualization is always achieved through the use of a *Mapping Mechanism (MM)* that maps virtual resources to real resources. The mapping mechanism is responsible for presenting the virtual resources as if they were real resources, and for maintaining the isolation between different virtual resources. We call this the *isolation* problem. The mapping mechanism also decides how resources are allocated; in other words deciding how to divide up or combine real resources to create virtual resources. This problem is known as the *embedding* problem. The embedding problem depends greatly on the isolation problem, since the method of isolation determines how the resources can be used. These two problems will be discussed in further detail later.

According to the authors of [29], and [30], the mapping mechanism is simply a function f that maps the set of virtual resources $\mathcal{V}$, to the set of real resources $\mathcal{R}$. The virtual resources can be thought of as the domain of f and the real resources as the codomain of f . The mapping function f maps each element in $\mathcal{V}$ to an element in $\mathcal{R}$.

$$f : \mathcal{V} \longrightarrow \mathcal{R} \cup \{t\}$$

such that if $y \in \mathcal{V}$ and $z \in \mathcal{R}$ then

$$f(y) = \begin{cases} z & \text{if } z \text{ is the real resource for virtual resource } y \\ t & \text{if } y \text{ does not have a corresponding real resource} \end{cases}$$

The value $f(y) = t$ causes a trap or fault handling procedure to occur by the mapping mechanism.

In all figures and examples until now, only one physical resource has been considered, represented as one real resource in the abstract domain. However, virtualization can apply to *multiple* real resources, that are representations of *multiple* physical resources. Similarly, there are several ways in which the mapping of virtual to real resources can be done - we identify four types of virtualization. In Figure 3.2 we show the general types of virtualization, where there can be multiple physical resources, multiple real and virtual resources, and multiple types of mapping.

The four types of mapping virtual resources, $\mathcal{V}_n$, to real resources, $\mathcal{R}_m$ are:

1. One-to-one: Mapping a single virtual resource to a single real resource, to allow for easier management of resources;

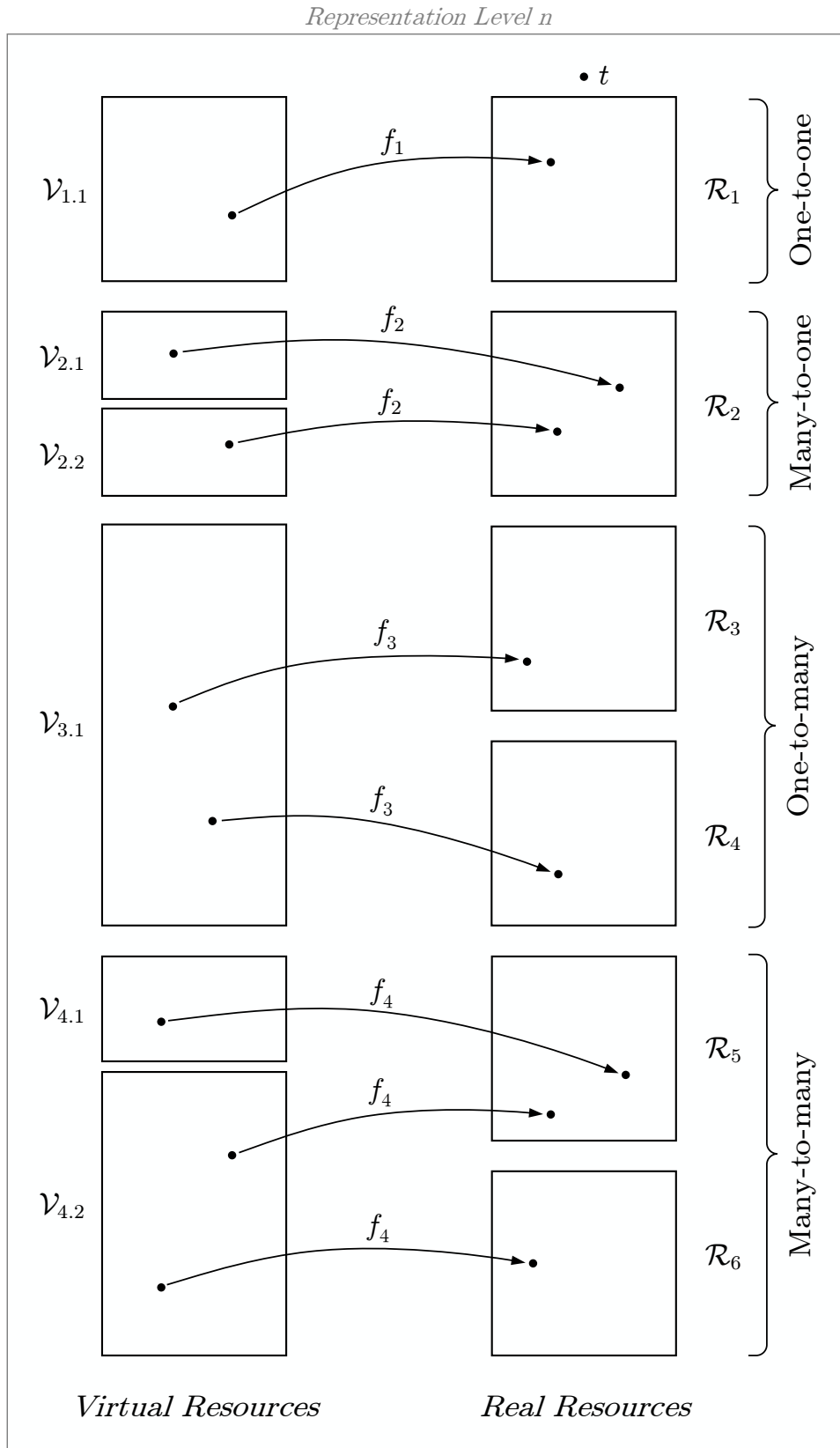


Figure 3.2: In this example, six physical resources are represented as real resources, $\mathcal{R}_{1-6}$, at some representation level n . Multiple virtual resources are mapped to the multiple real resources in different ways, showing the four types of virtualization. Many-to-many mapping is the ideal case as it enables abstract resources to be used in the most flexible manner.

2. Many-to-one: Mapping multiple virtual resources to a single real resource such as partitioning a single resource into a number of smaller and more easily accessible resources of same type;
3. One-to-many: Mapping one aggregated virtual resource to several real resources. Used to aggregate many individual components into larger resource pool; and lastly
4. Many-to-many: Mapping multiple virtual resources to multiple real resources. The combination of aggregating and partitioning resources to create completely customizable resources that can be tailored exactly to requirements.

Many-to-many can be considered the ideal case, since it enables resources to be used in the most flexible manner.

3.3 Recursion

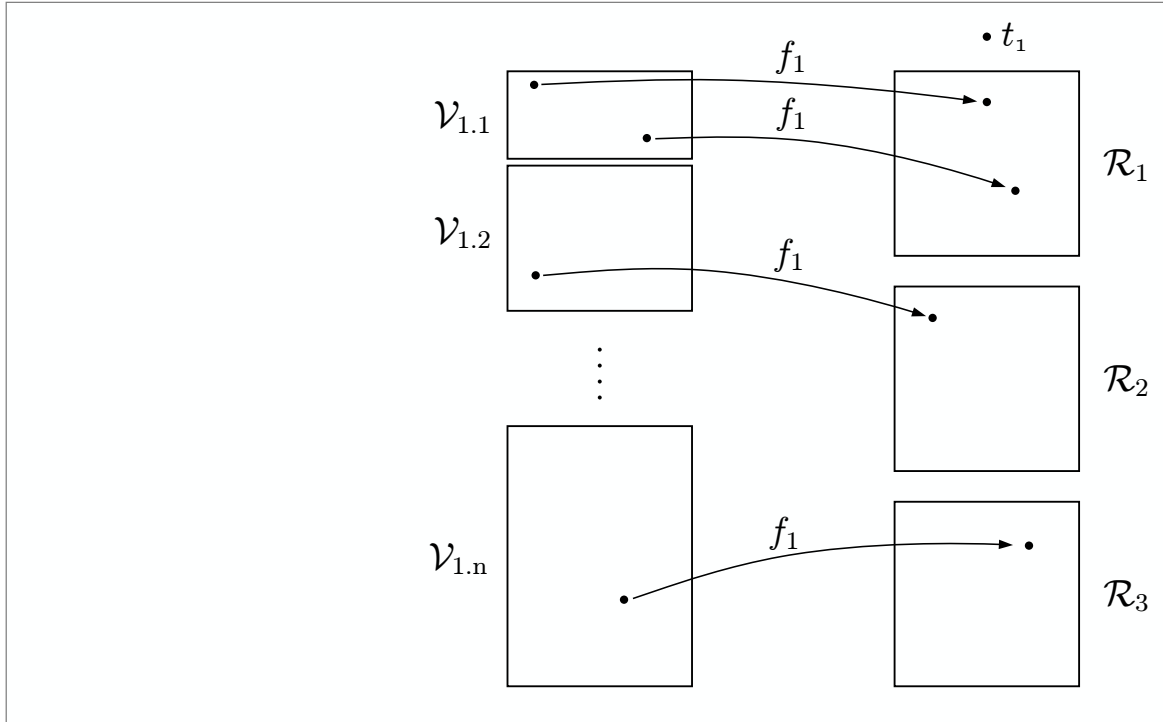
When virtual resources are presented in such a way that they are indistinguishable from real resources, recursion is possible [31]. Users could choose to virtualize their virtual resources, since they perceive them as real resources. The mapping of resources is now done twice, the first mapping maps the virtual resources received by the user to the real resources, and the second mapping maps virtual resources to virtual resources.

The terms ‘real’ and ‘virtual’ can be confusing when recursion is taken into account. For this reason we always refer to the codomain of the mapping function, i.e. the resources that are being virtualized, as ‘real’ resources, even though these resources might already have been virtualized by a previous virtualization instance.

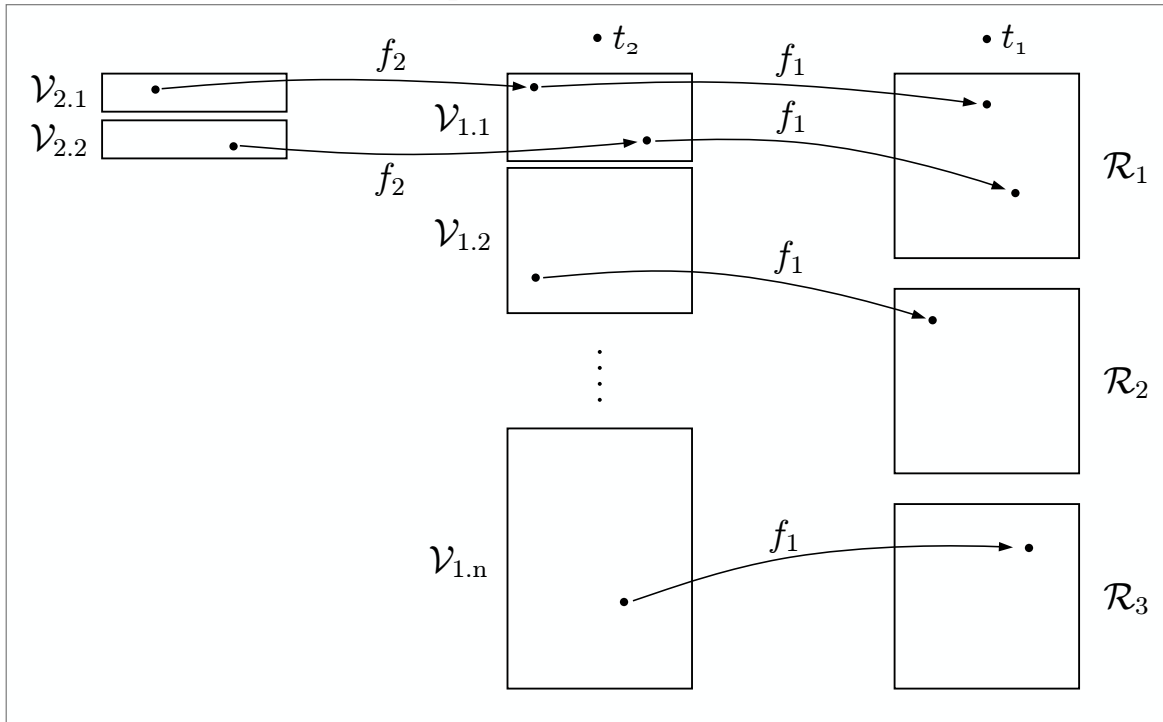
The resource mapping function, f , described above can be extended directly for recursion by applying the mapping function, f , multiple times and interpreting $\mathcal{V}$ and $\mathcal{R}$ as different instances of virtualization. Mapping function f_1 maps virtual resources $\mathcal{V}_1$ to real resources $\mathcal{R}$. Now, in a second virtualization instance, f_2 maps virtual resources $\mathcal{V}_2$ to ‘real’ resources $\mathcal{V}_1$.

$$\begin{aligned} f_1 : \mathcal{V}_1 &\longrightarrow \mathcal{R} \cup \{t_1\} \\ f_2 : \mathcal{V}_2 &\longrightarrow \mathcal{V}_1 \cup \{t_2\} \end{aligned}$$

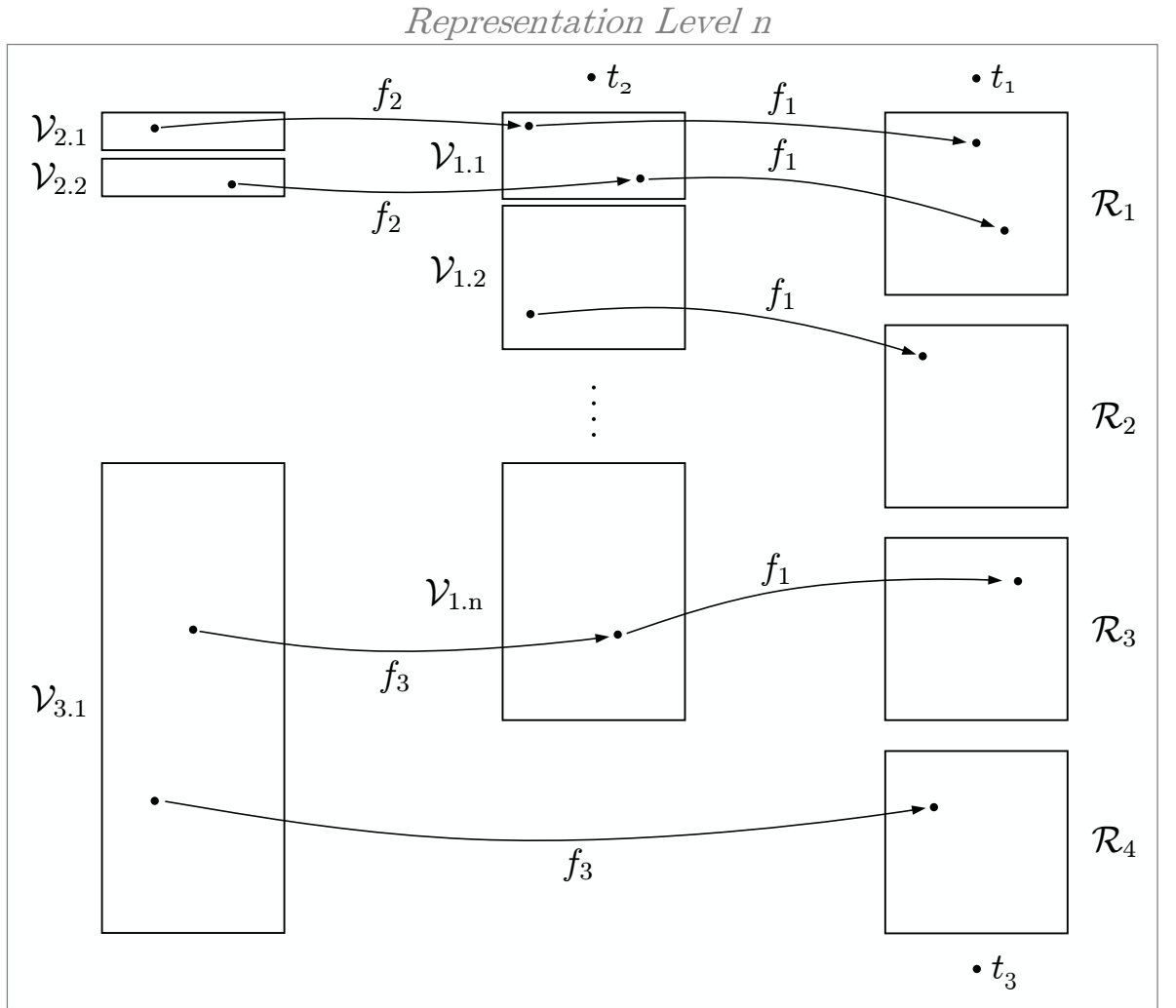
The real resources for f_1 are $\mathcal{R}$, and for f_2 the real resources are $\mathcal{V}_1$. Figure 3.2 illustrates several virtualization instances that recursively map virtual resources to real resources.

Representation Level n 

(a) In the first instance of virtualization, many-to-many mapping f_1 maps virtual resources $\mathcal{V}_{1.1-1.n}$ to real resources $\mathcal{R}_{1-3}$.

Representation Level n 

(b) Many-to-one mapping function f_2 is an example of recursive virtualization, since it maps virtual resources $\mathcal{V}_{2.1}$ and $\mathcal{V}_{2.2}$ to 'real' resource $\mathcal{V}_{1.1}$.



(c) The one-to-many function f_3 shows that it could be possible to map virtual resources to a combination of real resources, whether these have been previously virtualized or not. In this case $\mathcal{V}_{3.1}$ is mapped to $\mathcal{R}_4$ and $\mathcal{V}_{1.n}$.

Figure 3.2: Example of recursive virtualization. f_1 is the first instance of virtualization, f_2 is the second instance, and f_3 is the third. f_2 and f_3 are recursive virtualization, since they depend on the first instance of virtualization, f_1 .

We can say that recursion is a requirement for virtualization, since non-recursive virtualization is simply multiplexing [32]. In the case of perfect virtualization, i.e. that virtual resources can be used exactly as real resources and no overhead exists, infinite recursion is possible [32].

3.4 The Isolation Problem

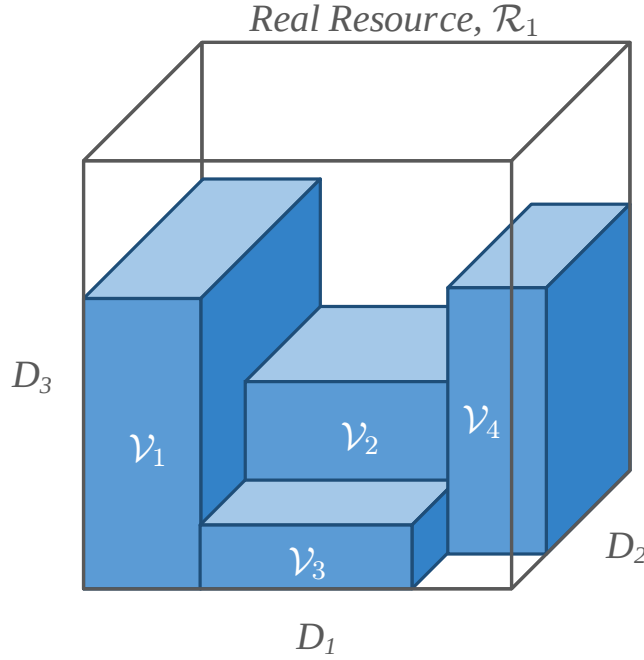


Figure 3.3: The real resource, R_1 , can be isolated along three dimensions, D_1 , D_2 , and D_3 . The users of virtual resources, $\mathcal{V}_1$, $\mathcal{V}_2$, $\mathcal{V}_3$, and $\mathcal{V}_4$, are only aware of and can only access each of their individual resources, which appear to them as real resources.

The isolation problem is the problem of choosing how to create virtual resources, and how to maintain independence between them. It should be impossible for VRs to interact with other VRs in any manner. By isolating along one or multiple dimensions of the real resources, each virtual resource user is only aware of its own virtual resources and can only use those resources, and thus it cannot interfere with other virtual resource users [33] [21]. This is illustrated in Figure 3.3. The term ‘dimension’ refers to a measurable feature of a resource.

Thus we add the isolation dimension(s) to the mapping function:

$$f_1 : \mathcal{V}_1 \xrightarrow{T} \mathcal{R} \cup \{t_1\}$$

for example using the time dimension. An example of how this applies is shown in Figure 3.4.

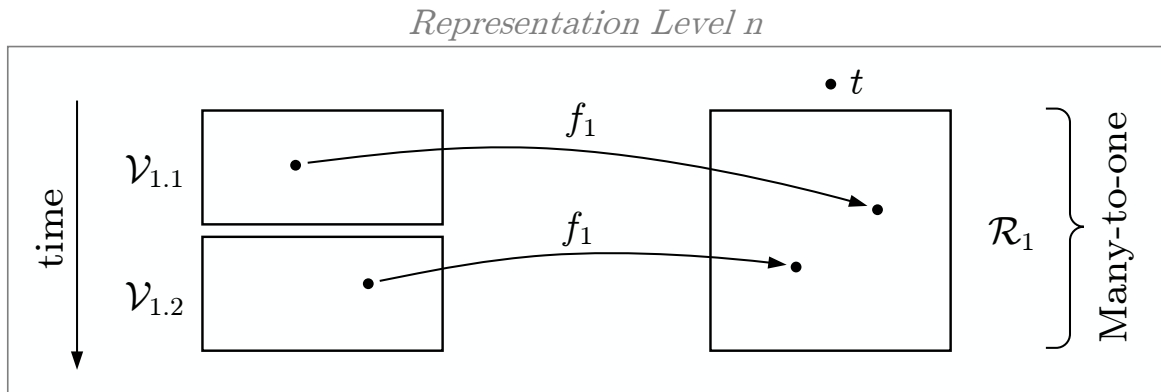


Figure 3.4: A many-to-one mapping using one dimension (time) for isolation.

However, the ability to isolate resources along a particular dimension depends on the technical capability of the mapping mechanism to measure, distinguish, and assign virtual and real resources at a suitable level of granularity, such that users do not perceive any difference between the virtual and the real resources. More specifically, the granularity of the mapping mechanism (in a particular dimension) should lie within a range of suitable granularities, depending on the **application** for which the user is using the resources, whether virtual or real. Therefore, it is not possible to give specific values for the range of suitable granularities, except within a specific context. However, in general, the upper limit of the range might be defined as the granularity at which the user notices that there is a difference in using the virtual and the real resources; while the lower limit might be defined by the granularity at which the resources can no longer be used effectively for the desired application. This should apply to both many-to-one (partitioning) and one-to-many (aggregation) virtualization.

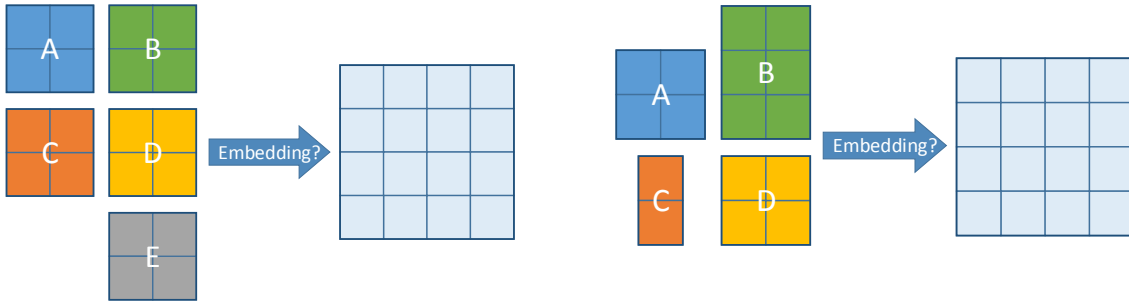
To give an example of the range of granularities, consider the virtualization of processing (or computation) resources. A real processor can be isolated in the time dimension to create virtual processing resources that are assigned to multiple users (many-to-one) in such a way that each user can use the processor in real-time. In this case, the upper limit of the range of granularities is the users' perception of real-time (which is human reaction time of about 0.2 seconds), and the lower limit is the time needed to perform a useful computation (which depends on the processor but is in the order of nano seconds today). Therefore, as long as the granularity used by the mapping mechanism is in the range $0.2\text{--}1 \times 10^{-9}$ seconds, the resources can be isolated successfully.

In summary, the isolation problem relies on the use of suitable granularities in each dimension, and that the mapping mechanism is technically capable of accurately employing these granularities. The isolation problem is a prerequisite to tackling the embedding problem, and influences the embedding problem greatly.

3.5 The Embedding Problem

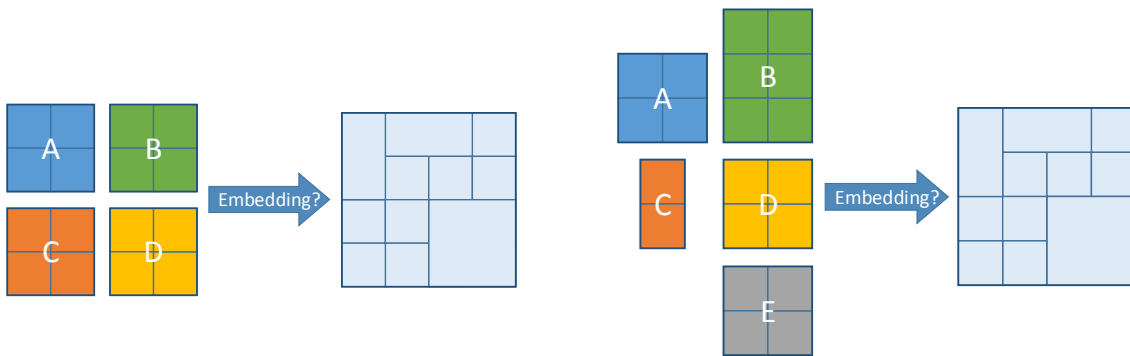
The problem of deciding how to map virtual resources to real resources is also known as the embedding problem. Essentially this is a resource allocation problem, which is the distribution of scarce resources to competing users. There are several problems that can be considered:

1. The first case is that there might not be enough resources to satisfy all of the users' requests;
2. The second case is that the users can request different quantities of resources; and



(a) The first type of problem occurs when there are more user requests than resources. At least one user will not receive resources.

(b) Another type of problem can happen when users have different request sizes.



(c) A similar problem can occur if the resources are of different sizes.

(d) The last example shows that these three problems can occur simultaneously.

Figure 3.5: This figure shows resource allocation problems that can happen when users A, B, C, D, and E request two-dimensional resources from a resource owner. There are three different types of problems that can occur, shown in a), b) and c). The three can occur simultaneously as shown in d). Resources can be allocated request by request in sequence, or by considering multiple requests together, which could provide more efficient embedding.

3. The third case is that each of the resources can be of unequal value.

These three problems are not mutually exclusive and often occur simultaneously. Depending on the objective(s) that the resource owner wants to achieve, different metrics can be used to determine the optimal resource allocation. Even when there are enough resources to satisfy all of the users' requests and a potential solution exists, resource allocation can be a complex problem to solve.

In the context of virtualization, users can make requests for virtual resources and an embedding algorithm determines which requests are successful. Figure 3.5 shows the resource allocation problems that can occur when users make requests for two-dimensional sets of resources.

3.6 Definition of Virtualization

Now that we have a better understanding of virtualization, we propose the following definition for virtualization:

Virtualization is a resource mapping that occurs in the abstract domain. Virtualization takes place within any one representation level in the abstract domain. The representation level used is a design choice. Abstract resources before virtualization, known as real resources, are limited by the granularity of the underlying physical resources. Abstract resources after virtualization, known as virtual resources, are not, and can be larger or smaller than real resources. Virtual resources are independent and can be allocated simultaneously to multiple users, each with the illusion of full ownership of their resources. Virtualization can always be done recursively, since users perceive no difference between the virtual resources and the real resources.

This definition is satisfactory since it addresses the concepts of 1) abstract resources as representations of physical resources, 2) the splitting and combining of resources, 3) ownership and isolation, 4) allocation, and 5) recursion.

3.7 Validity of the Theory

Although we have attempted to validate virtualization theory as much as possible by referring to previous works, some aspects are new and need verification. One method

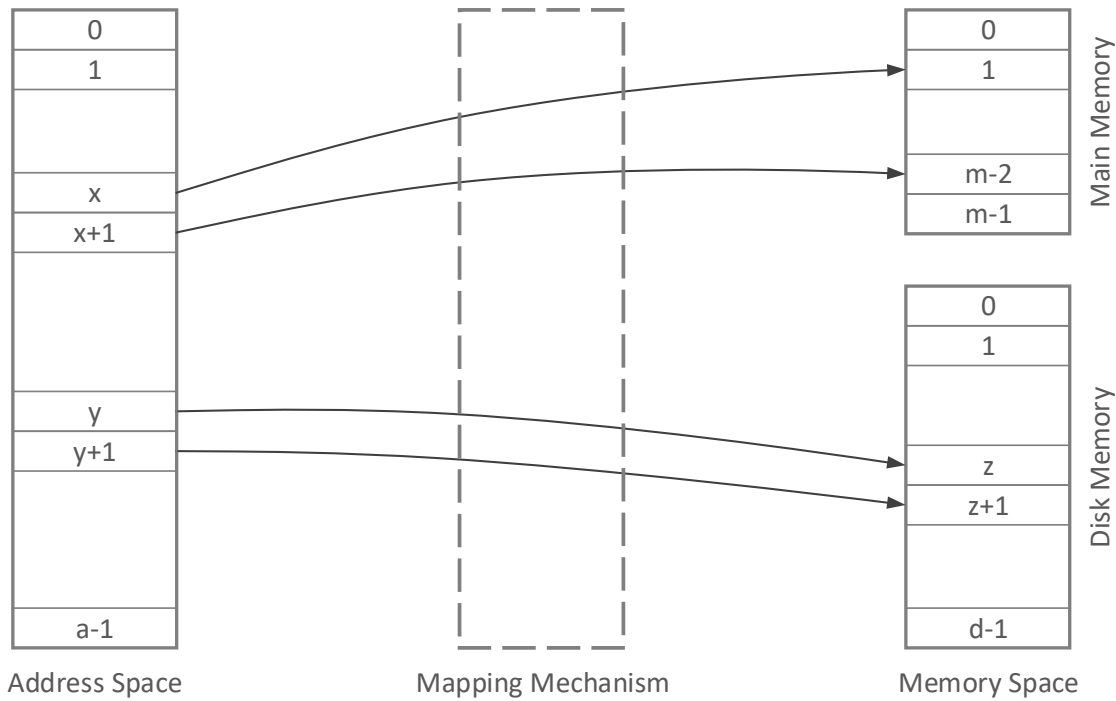


Figure 3.6: The separation of address and memory spaces allows the address mapping mechanism to combine main memory and disk memory into a one-level storage space. The address mapping mechanism can also give users the perception of having a unique address space for multitasking

of verifying the theory is to examine virtualization technologies and to see if the theory holds. As an example we examine virtual memory, one of the first virtualization techniques developed.

Virtual memory, or one-level storage as it is also known, was developed in 1961 by the Atlas group to overcome the storage allocation problem of distributing information between main memory and auxiliary memory levels in computers [34]. In a one-level storage system, a distinction is made between the address space, which is the set of identifiers used to refer to information, and the memory space, which is the set of physical memory locations used to store information [35]. Instead of offering programs direct access to the memory space, programs are only offered access to the *address* space, and a supervisor maps the address space to the memory space. By decoupling the address space in this way from the physical memory, it is possible to *combine* both main memory and auxiliary memory into a single address space, thus offering the illusion of one-level storage, shown in Figure 3.6.

Let us now examine how virtualization theory applies to virtual memory, summarized in Table 3.1. Firstly, it is important to identify the logical resources that are being virtualized. In this case the logical resources are information in the form of bits. Next, it

is possible to see that virtual memory is a one-to-many mapping, since it maps one virtual resource (the address space) to many real resources (the main and auxiliary memory spaces)⁴. The isolation dimension is the address space, which is to say the index of locations to store information. In theory recursion is possible in virtual memory, because the resources offered are locations to store information, exactly the same type of resources as non-virtualized memory. In fact the authors of [36] develop such a recursive virtual memory system. Some examples of different approaches to the embedding problem in virtual memory, are the different paging and segmentation algorithms that have been developed [30].

Table 3.1: Virtualization theory example: Virtual memory

Concept	Use in Virtual Memory
Abstract resources at a representation level	Location to store bits
Mapping	One-to-many / Many-to-many
Isolation	Address space - index of storage locations
Recursion	Possible but not very useful in practice, see [36]
Embedding	Paging algorithms, segmentation algorithms

The example of virtual memory shows how virtualization theory can apply in practice. We see that each aspect of the theory has a practical counterpart in virtual memory. This offers some validation of the theory. Later we will examine several other virtualization technologies in the context of wireless networks, but for each virtualization technology the same analysis could be performed.

3.8 Summary

In this chapter, we formulated a theory for virtualization and used the theory to generate a precise definition of virtualization. Although inspired by previous works, many aspects of the theory do not exist in the literature (to the best of our knowledge), such as 1) the description of virtualization in terms of representation, 2) the four types of virtual resource mapping, 3) a consistent description of recursion, 4) the identification of the isolation and embedding problems as separate problems, and most importantly 5) a precise definition of virtualization. We also provided some validation of the theory

⁴With the development of multitasking, in reality virtual memory is now many-to-many, because different programs each have their own address space.

by applying the theory to the example of virtual memory. The theory will come into use in future chapters, firstly by allow us to describe the different forms of resource virtualization coherently, and secondly by enabling us to perform a survey of wireless link virtualization with clarity.

4 Virtualization in Wireless Networks

4.1 Introduction

Having developed a theory of virtualization, we now turn to the second research question of the thesis: how can we use our precise theory of virtualization to do research on wireless network virtualization?

To answer this question, we first describe the development of virtualization in wireless networks in this chapter, before surveying the existing literature in the following chapter.

We define a wireless network as ‘a set of nodes that can transfer information through links, where some of the links may be wireless in nature’. From this definition, we can deduce that wireless networks consist of two parts: nodes and links. Before we can virtualize wireless networks, we must first be able to virtualize nodes and links individually. Thus as a first step, we examine what constitutes nodes and links, and analyse how node and link functionality can be virtualized.

4.2 Nodes

In this thesis we consider a node as a physical device (i.e. computer hardware) that can determine the outcome of abstract operations through physical manipulations [23]. We identify four specific functions for any node: *input* (I), i.e. the instantiation of abstract objects in a physical state, *storage* (S) of that physical state in some way, *processing* (P) of the physical state in a way that is commutative to some form of abstract operations, and *output* (O) of the result of the physical process in the form of an abstract representation. All four functions are required within a node, since, for example, it would be pointless to have processing available but no input, as there would be no information to process. Figure 4.1(a) relates the four functions to the commuting diagram of computer hardware

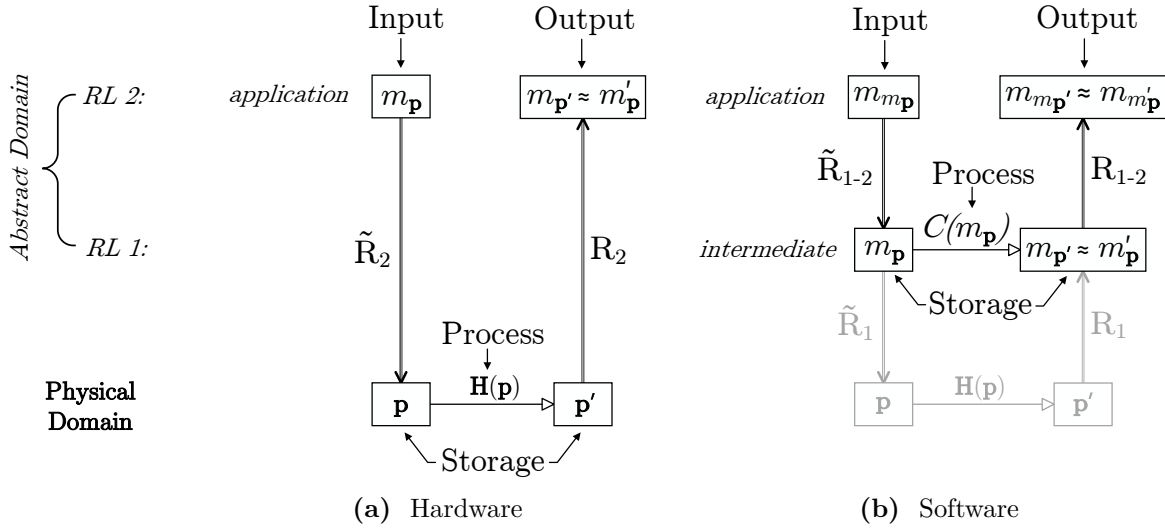


Figure 4.1: (a) Model of computer hardware based on AR theory. Computer hardware can be thought of as having four functions, corresponding to the mappings of AR theory and storage of the physical states. (b) Software must be used to enable all four functions to be virtualized, i.e. performing the compute cycle in the abstract domain, as virtualization can only be done in the abstract domain.

from Abstraction/Representation (AR) theory. These four functions are consistent with the Input-Process-Output and Storage (IPO+S) model of computing [37].

Referring back to section 2.5 in Chapter 2, remember that the compute cycle can also be done in software. In this case the representation/instantiation occurs completely in the abstract domain, and the four functions of the IPO+S model are all in the abstract domain, shown in Figure 4.1(b).

Virtualization allows physical resources to be used more efficiently, and can provide new functionality. However as mentioned previously, virtualization can only be done in the abstract domain. Thus to virtualize node functions, these functions must be done in the abstract domain, meaning in software.

Each function (input, storage, processing, output) can be virtualized individually, and full node virtualization (known as machine virtualization) can only be considered when each type of function can be virtualized⁵. Therefore we briefly discuss processing-, storage-, and Input/Output (I/O) virtualization below. We refer to physical resources according to their function type, for example we refer to storage resources rather than resources that instantiate storage functionality. Although technically incorrect, this simplification makes it much easier to follow.

⁵Machine virtualization was actually developed before input and output virtualization, but had problems when multiple virtual machines used I/O simultaneously.

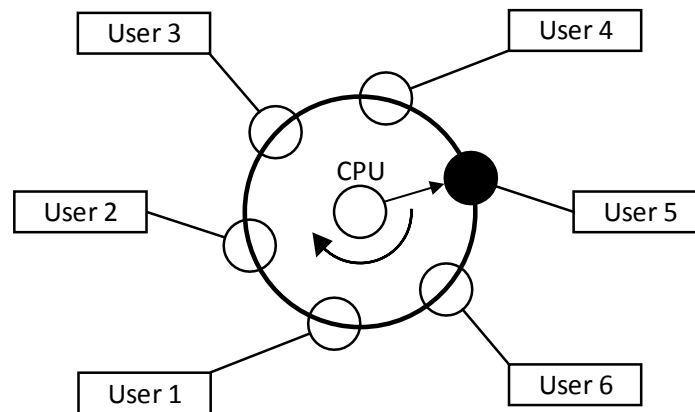


Figure 4.2: Time-sharing gives several users the perception of having exclusive use of a processor, by giving each user a short burst of computing time. In this example, the CPU is processing user five's burst of time, before it moves on to the next user.

4.2.1 Process Virtualization

The idea of process virtualization can be traced back to the concept of compatible time-sharing, first developed at M.I.T. in the early 1960's [38]. Time-sharing was developed to overcome the limited man-machine interaction of batch processing, which had led to an increase in programming errors and debugging time, as larger and more complex programs were being set [39].

Time-sharing enables several people to make use of a computer at the same time, shown in Figure 4.2 [40]. Rather than offering users direct access to computing resources, which can lead to serious crashes and memory problems, a supervisor buffers user input, and sequentially runs user programs for small bursts of time. The full sequence of user programs occurs frequently enough (ideally in less than ~ 0.2 seconds) that a computer appears to be fully responsive to all users.

By mapping user processing time to bursts of machine processor time in this way, and by maintaining strict isolation, users have the perception of exclusive use of dedicated processors. Thus the illusion of multiple 'virtual' processors is created. Thus virtualization both increases resource efficiency, and also offers users new or improved functionality. The surveys [41–43] provide further information about processing virtualization techniques, and a very interesting and informative documentary on the compatible time-sharing system can be found at [44].

4.2.2 Storage Virtualization

One method of storage virtualization is virtual memory, which we discussed in section 3.7. The use of virtual memory not only automates the storage allocation problem efficiently [45], but also enables machine independence, program modularity, convenient memory addressing, and the capability of handling structured data [35], [30]. For a more detailed view on storage virtualization techniques, see the surveys [46], and [47].

4.2.3 Machine (Node) Virtualization

The development of process and storage virtualization led to new functionality, and computing became thought of as a large system of components serving a community of users, where each of the users could run different programs with different processing, memory, and I/O interaction requirements [48]. In such a system, software is commonly split into two classes to avoid system integrity issues: a privileged supervisor (or Operating System) which is presumed to be correct, and a second non-privileged class which is denied any functionality that can cause interference between processes [49].

However, this arrangement only allows one privileged supervisor to be run at a time, and incompatible non-privileged programs cannot be run easily [50]. Machine virtualization overcomes this problem by constructing simulated copies of the machine, known as virtual machines, and each virtual machine can run a different privileged supervisor [51]. A Virtual Machine Monitor (VMM), also known as a hypervisor, isolates process and memory operations for each virtual machine, and maps them to the host machine using time-sharing and virtual memory techniques [52]. Until recently I/O operations had to be trapped and executed by the VMM.

Advances in machine virtualization, especially in server virtualization have reduced the cost of servers hugely and has led to the widespread adoption of moving computing tasks to the ‘cloud’ [53]. The recent development of I/O virtualization (see next part) has allowed full computer node virtualization, consisting of storage, process and I/O virtualization. There are many contexts in which the use of node virtualization is growing such as desktop, application, and user virtualization. More information can be found in the works [52], [54], [55], [56], and [57].

4.2.4 Input and Output Virtualization

One of the problems encountered by early virtual machines was the mapping of Input/Output paths from virtual device addresses to real device addresses, since absolute addressing is required for I/O paths [50]. Early VMMs trapped I/O instructions used by the virtual machines, copied instructions, and ‘absolutized’ them by mapping the virtual addresses to the correct real I/O addresses. While this solution of emulating I/O devices in software enabled virtual machines to use I/O operations, it was a work-around and not I/O virtualization.

The generalized I/O memory management unit (IOMMU), developed by Intel, is a device that maps virtual addresses to real ones across isolated partitions, supervised by system software [58]. The VMM controls these partitions and thus full virtualization of I/O operations is possible, since virtual I/O operations are mapped directly to the devices. One example of an I/O peripheral that has been virtualized is the Network Interface Card (NIC). A survey of I/O virtualization techniques is given in [59].

4.3 Links

The function of links is to transfer information between nodes in a reliable manner. Similarly to nodes, links can also be thought of as consisting of abstract objects (such as information) and physical resources which instantiate that information and physically send it between nodes. Again, similar to nodes, links can only be virtualized in the abstract domain.

Links can be wired or wireless. However, there are significant differences between wired and wireless links, due to the nature of the physical resources used. Both wired and wireless links represent information using the electromagnetic spectrum, but in the case of wired links, the electromagnetic spectrum is isolated from other links through the use of physical cables. For wireless links however, additional measures must be taken to provide isolation and reliability since all links are broadcast.

4.3.1 Wired Link Virtualization

Wired link virtualization developed as an indirect consequence of time-sharing. Time-shared systems allowed users to work from remote terminals connected to a mainframe computer, either using dial-up lines over the Public Switched Telephone Network (PSTN),

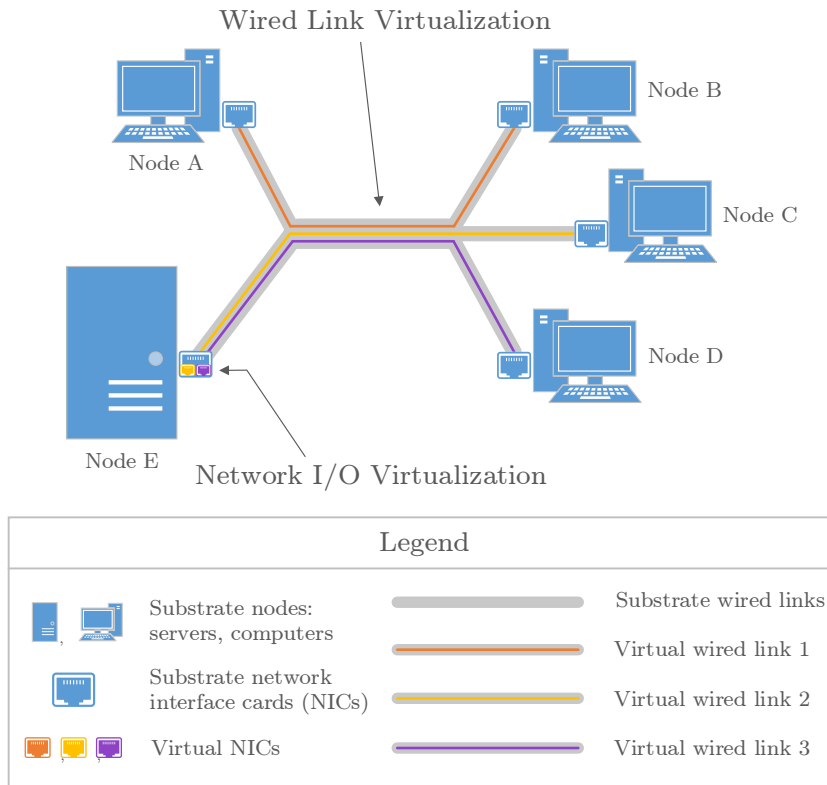


Figure 4.3: In this network, virtual link 1 can exist without network I/O virtualization, while network I/O virtualization at node E is needed to enable virtual links 2 and 3 to exist. Although it is possible to achieve wired link virtualization without network I/O virtualization (NIC virtualization), the full benefits of link virtualization cannot be gained without network I/O virtualization.

or through private lines leased from the PSTN operators. The dial-up lines were much cheaper to use, but suffered from lack of security and functionality, while private lines offered good security and functionality but were expensive.

Virtual private network (VPN) services offer the security and functionality of private lines at much cheaper costs by exploiting the fact that typically communication between nodes in a network only occurs for a small percentage of time. Thus physical links can be time-shared to provide the illusion of private links, known as virtual circuits [60]. VPNs can be scaled much easier than physical private links, and can also be tailored to suit users' preferences [61].

However, although VPNs can isolate different logical networks over a shared infrastructure, they are prone to some limitations. Among these are that the coexistence of different networking solutions is not possible, and also that virtual networks are not fully independent [62]. Another limitation is that broadcasting is not supported in the same way as on native networks. A final limitation is that additional security measures are needed for VPNs which can add overheads, and these and other virtualization overheads

can decrease the network reliability, throughput and latency.

Therefore, new methods of wired link virtualization are being developed and wired link virtualization is an active research topic. The survey [63] describes several link virtualization techniques, while the work [64] describes link virtualization for optical networks.

I/O virtualization and link virtualization

It is possible to use link virtualization without network I/O virtualization (network interface card virtualization in wired networks), such as when multiple virtual links share a real link but each node only has one virtual link. However, without network interface card virtualization it is not possible to have multiple virtual links to a node over one real link, or on the contrary, one virtual link to a node composed of multiple real links [65]. Therefore, network I/O virtualization is required to obtain the full benefits of wired link virtualization, as shown in Figure 4.3.

4.3.2 Wireless Link Virtualization

Although wired link virtualization has been studied for many decades, Wireless Link Virtualization (WLV) is a developing research area. As mentioned before, wireless link virtualization is the main focus of this thesis. Therefore we briefly introduce wireless link virtualization here, whereas a detailed examination of WLV follows in the remaining chapters of this thesis.

Wireless link virtualization is the process of virtualizing wireless links, creating virtual resources that are isolated and which can use differing technologies and/or configurations independently. Although appearing to be similar to wireless resource sharing, wireless resource sharing is fundamentally different to virtualization, since wireless resource sharing does not create independent resources. In addition, virtualization allows the combination of resources to occur, which resource sharing does not.

There are several complications which exist in wireless link virtualization which do not exist in wired link virtualization due to the difference between the wired medium and the wireless medium. The first difference is that because of the inherent variation of the wireless channel over time, it is not possible to predict the information throughput of wireless links in advance. A second difference is that wireless links are broadcast, and thus have the potential to interfere with any other wireless link, whereas in wired links this does not occur [33]. Another complication for wireless links is that wireless

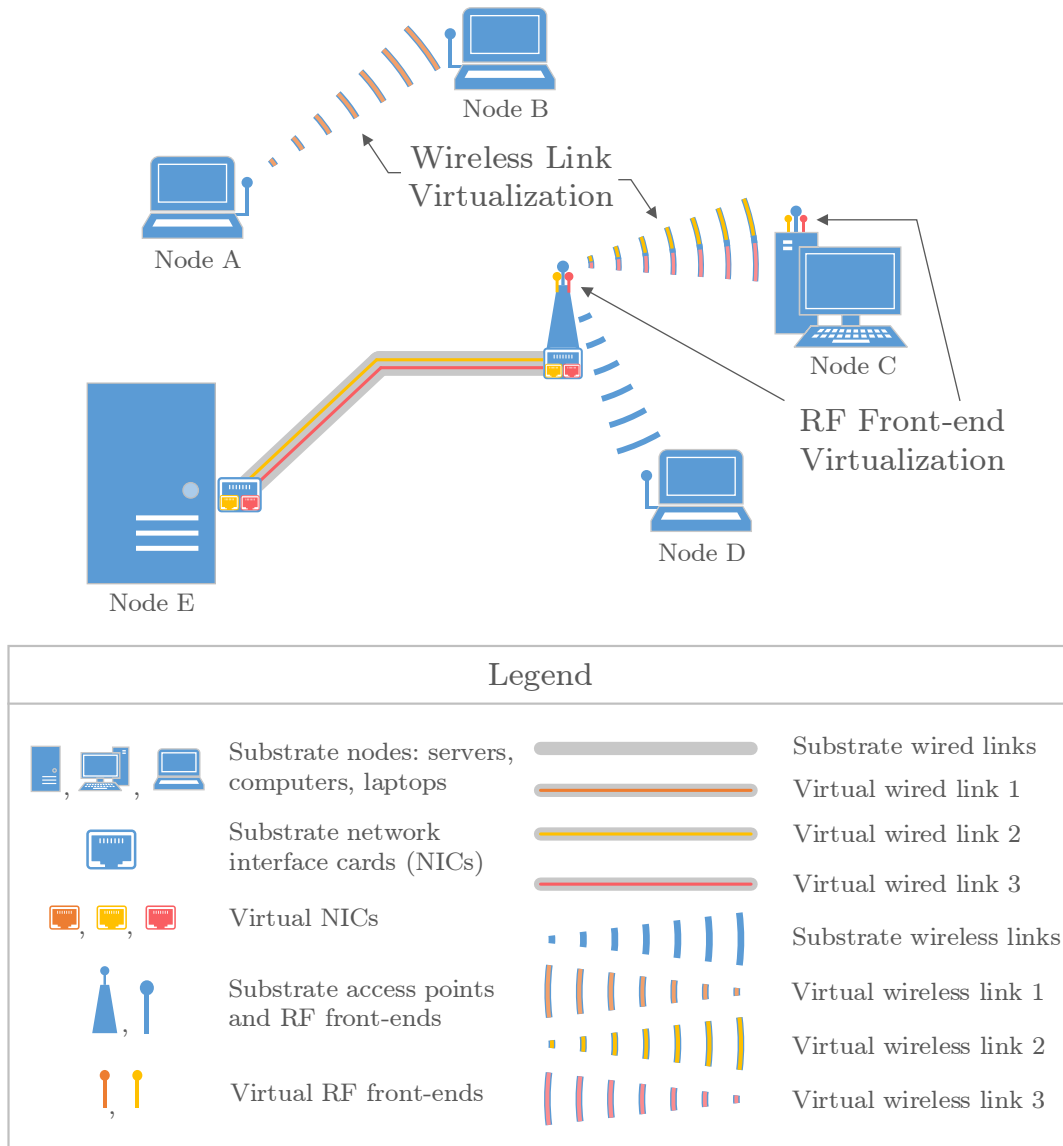


Figure 4.4: Building on the previous figure, wireless link virtualization is used to create a customized, flexible virtual link between node A and node B. However, Radio Frequency (RF) front-end virtualization is needed, combined with wireless link virtualization, to create multiple virtual links between node C and node E .

nodes tend to be highly mobile, which means it is harder to predict and provision for the information transfer between nodes.

Because of the above reasons, WLV is a difficult problem and although there has been a significant amount of work on WLV in recent years, to date there has not been a satisfactory solution. In the next chapter, we perform a survey of wireless link virtualization, aided by virtualization theory, which allows us to identify the main research problems in WLV.

I/O virtualization and link virtualization

Similarly to wired link virtualization, it is possible to use wireless link virtualization without virtualizing the network I/O, which in the wireless case consists of the Radio Frequency (RF) front-end. However, without RF front-end virtualization, it is not possible to use multiple virtual links per real link, or the inverse, which massively decreases the potential applications of wireless link virtualization. An example of wireless link virtualization with and without RF front-end virtualization is shown in Figure 4.4.

4.4 Wired and Wireless Network Virtualization

Having examined the virtualization of the resources that make up networks, we now examine wired and wireless network virtualization.

4.4.1 Wired Network Virtualization

The idea of [wired] Network Virtualization (NV) has arisen as a solution to several problems with the Internet today. As a result of the Internet's success and ubiquity in many areas of life, it has become subject to ossification, since the competing interests of existing stakeholders and the large capital investment needed have led to high resistance to disruptive technologies [66], [67]. The need for global agreement between competing providers has limited innovation to simple incremental updates, which do not satisfy demand for new services and functionality, or to ad-hoc workarounds, which do satisfy legitimate needs, but violate core Internet design principles, and as such have impaired flexibility, security, reliability, and manageability [68], [62].

Network virtualization has evolved from the concept of using overlay networks to address these ossification issues. Initially, it was envisaged that overlay networks would be highly programmable platforms for innovation, while simultaneously allowing the existing network infrastructure to be maintained [69], [66]. However, later it was realized that overlay networks could not provide the innovation and flexibility desired, and that it would be necessary to virtualize the underlying network infrastructure [68].

Network virtualization can be considered as using both node and link virtualization to create complete virtual networks (sometimes called Meta-Networks) [70], [67]. Figure 4.5 shows the distinction between virtual nodes, virtual links and virtual networks. In addition to offering a potential solution to the Internet impasse, network virtualization also enables the separation of existing Internet providers into the roles of Infrastructure

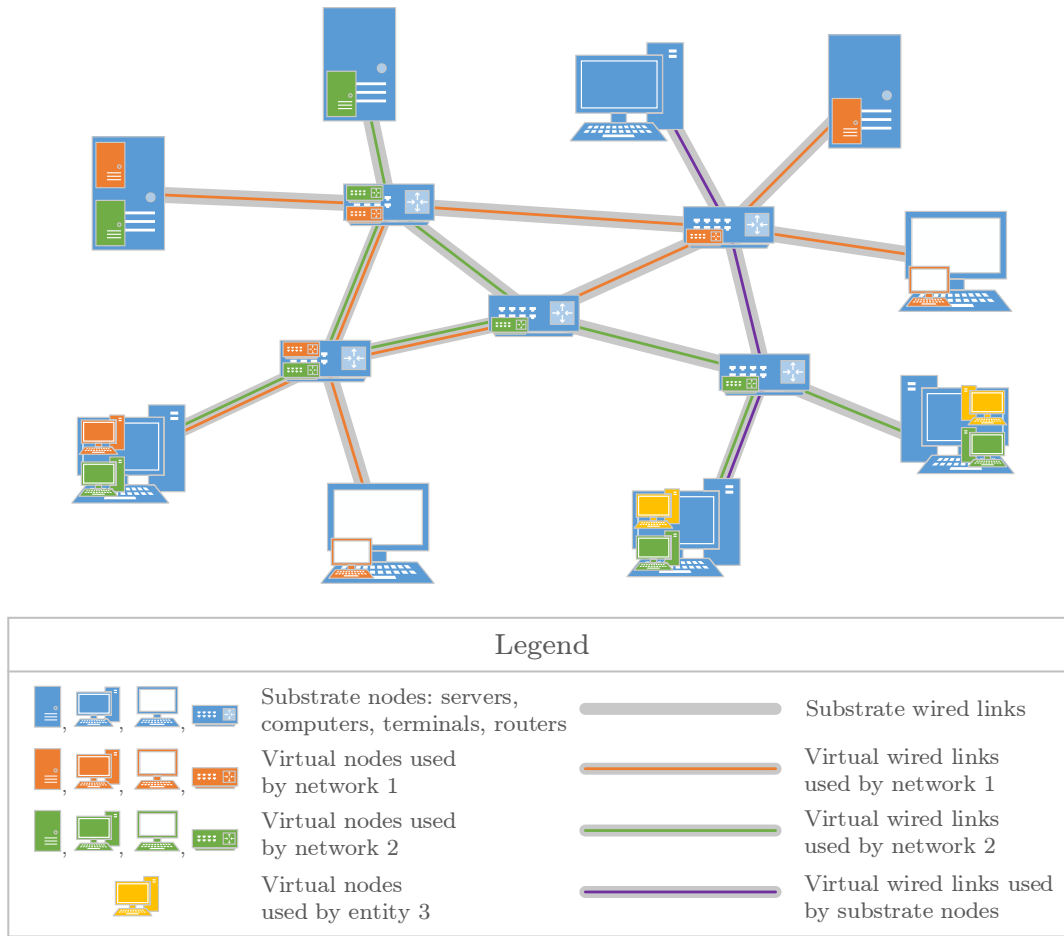


Figure 4.5: Wired network virtualization: Virtual networks A (orange) and B (green) can co-exist simultaneously on the underlying substrate. These networks can offer customized services to their users in a secure and isolated manner. Another entity C requires several computing nodes, but no network connectivity and so node virtualization provides entity C with virtual nodes (yellow). Meanwhile, virtual links or VPNs (purple), can be used to connect several substrate nodes. Note that for the sake of simplicity, we do not show network I/O virtualization in this figure but assume it is used.

Provider (InP) and Service Provider (SP) [71]. As argued in [67] and [71], there are several reasons why this is beneficial:

1. Existing network providers have very few opportunities to distinguish themselves from their competitors and thus they equally have little incentive to develop and deploy new solutions. Decoupling InPs and SPs enables diversity and innovation, leading to better infrastructure and new and improved services for end users.
2. Separating the role of infrastructure and service providers lowers the barriers of entry significantly, since SPs do not need to invest in their own equipment. This leads to increased competition and innovation.
3. A separation of the two roles enables the sharing and aggregation of network in-

frastructure, leading to increased efficiency and cost savings.

4. In the current network model, new software and network protocols cannot easily be tested, and often not under real traffic conditions. By splitting the roles of SP and InP, new experimental software and protocols can be tested in isolated networks, without affecting existing services. The isolation of networks also enables SPs to customise their networks and achieve better security.

Because of the many potential benefits that can be obtained, network virtualization is an area of ongoing research, and several NV testbeds have been constructed. Initially, testbeds have focused on node virtualization such as PlanetLab [66] and GENI [1], but more recently, testbeds such as CABO [71] and 4WARD [3] are interested in full network virtualization, i.e. both node and link virtualization combined. There is also significant ongoing research on network virtualization, summarized in the surveys [72] and [6].

Two interesting ongoing research topics which are related to network virtualization are Software Defined Networking (SDN) and Network Function Virtualization (NFV). Although these topics are outside the scope of this thesis, they are discussed briefly.

Software Defined Networking

Software defined networking is generally defined as “the separation of [network] forwarding hardware from the control logic” [73]. What this means in practice is that network resources, such as switches, routers, and other types of middleboxes, can be configured dynamically through software, rather than previously being configured in a static manner embedded in hardware. Therefore, SDN allows networks to become more *programmable*, more *open*, and more *extensible*, leading to many benefits [73]. Some of these benefits are: easier management of network resources, dynamic reconfiguration and reallocation of network resources, centralised network management (termed “decoupled control”), and faster development and deployment of network technologies and protocols [74]. One of the most prevalent use-cases of SDN is to decouple the data plane and the control plane, meaning that distributed network resources can be reconfigured dynamically by a centralised controller.

In the context of AR theory, SDN can be viewed as a specific instance of moving from hardware-based computing, to software-based computing (see Figure 2.6 in Chapter 2). As shown in Figure 4.6, we can think of SDN as moving application functionality, in this case data forwarding and control logic, from an embedded hardware platform, to a software-based platform that can be reconfigured dynamically (for the sake of simplicity, the figure omits the full compute cycle and only shows the various representation

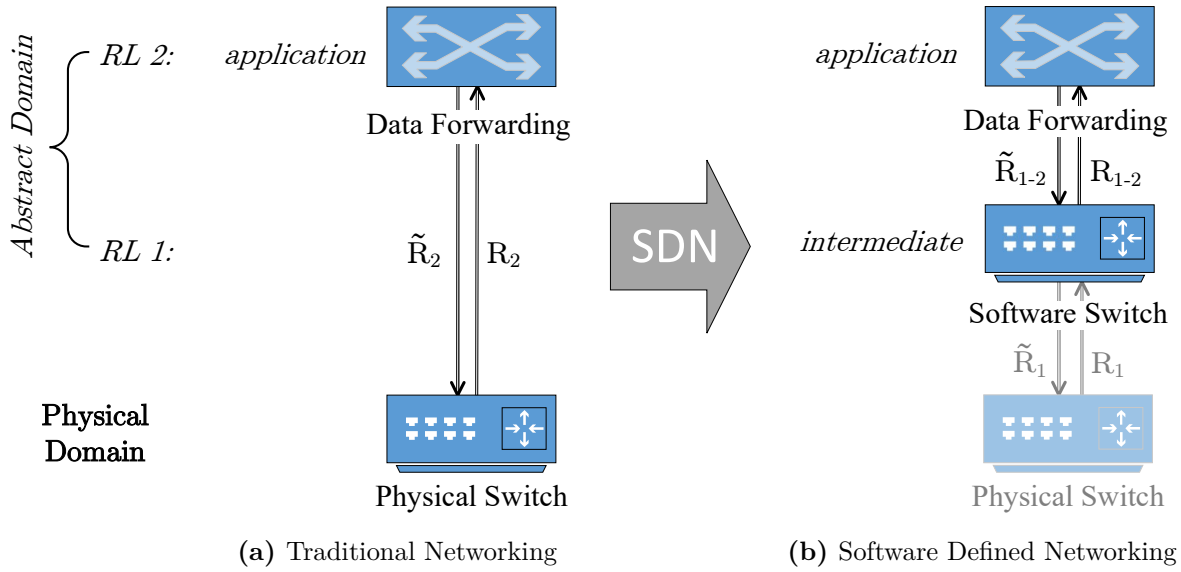


Figure 4.6: Software defined networking (SDN) adds an intermediate representation level between the application representation level and the physical hardware for networking resources such as switches. In this way, the functionality performed by networking devices, such as data forwarding and control logic, can be reconfigured dynamically, and the resources can be used in a more flexible manner. One of the most prevalent use-cases of SDN is to “decouple” the data plane and the control plane, meaning that distributed network resources can be reconfigured dynamically by a centralised controller.

levels). Through the addition of an additional level of representation – the intermediate or software representation level – SDN allows network resources to be used in completely new ways.

In addition to the benefits already mentioned, another huge advantage of SDN is

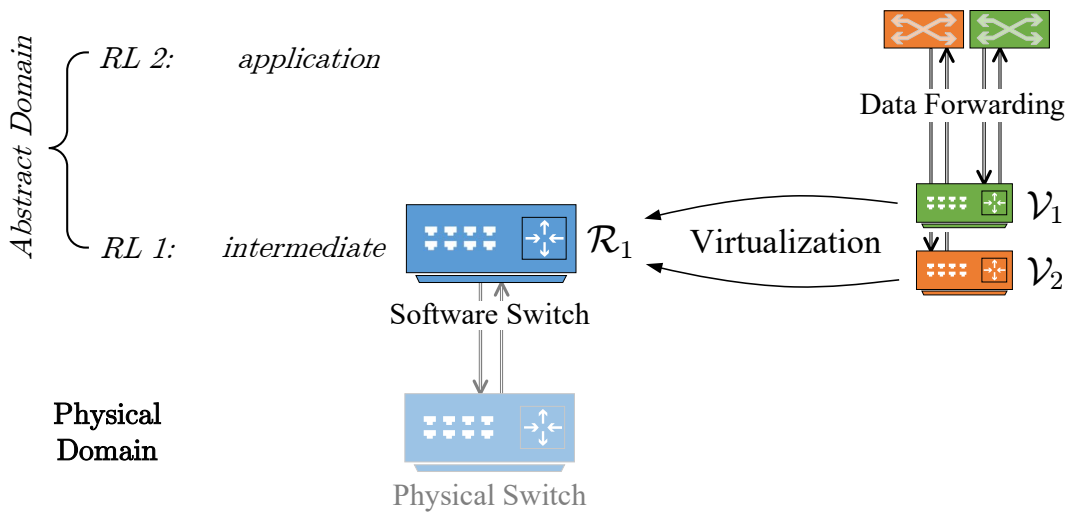


Figure 4.7: SDN is a prerequisite for network resource virtualization, as it moves the compute cycle to the abstract domain.

that since the compute cycle is now completely done in the abstract domain rather than between the abstract and the physical domains (see Chapter 2), virtualization of network resources is possible. Therefore SDN can be viewed as a prerequisite for network resource virtualization, as shown in Figure 4.7.

Network resource virtualization, i.e. the virtualization of switches, routers, and other middleboxes, is a specialised form of node virtualization, necessary to enable network virtualization. Therefore SDN is extremely important for future networks – both wired and wireless – which will rely on network virtualization to provide new services and better efficiency [7].

Network Function Virtualization

Traditionally in networking, certain network services are broken down into several network functions, each of which performs a specific task. These network functions are commonly run on specialised hardware, optimised for performance (such as latency, throughput, etc.). There are advantages to breaking network services into network functions in this manner, as it enables higher performance and modularity, leading to easier management and scalability.

However, as the number of network services continues to grow, and as new technologies (that require new network functions or new implementations of existing network functions) are being deployed, managing these dedicated hardware nodes is becoming more and more difficult [75]. Therefore, Network Function Virtualization has been proposed as a potential solution in a white paper by The European Telecommunications Standards Institute (ETSI) [75]. By implementing network functions on commodity hardware through software virtualization techniques, it is hoped that a) network functions will be much easier to manage and scale, b) network functions can be provisioned faster with more flexibility, c) equipment and energy costs can decrease, d) new network functions can be developed faster and more reliably through better testing, and e) network functions can be standardised more easily, enabling interoperability between different networks [76], [77].

From the perspective of the framework developed in this thesis, NFV can be considered as a form of node virtualization, albeit specifically tailored to networking. Similarly to SDN, the transformation of network functions running on dedicated hardware to virtualized network functions relies first on the use of an intermediate representation level, as shown in Figure 4.8. Subsequently, the intermediate software can be virtualized and used for running network functions such as firewalls, deep packet inspection (DPI), and network address translation (NAT), etc.

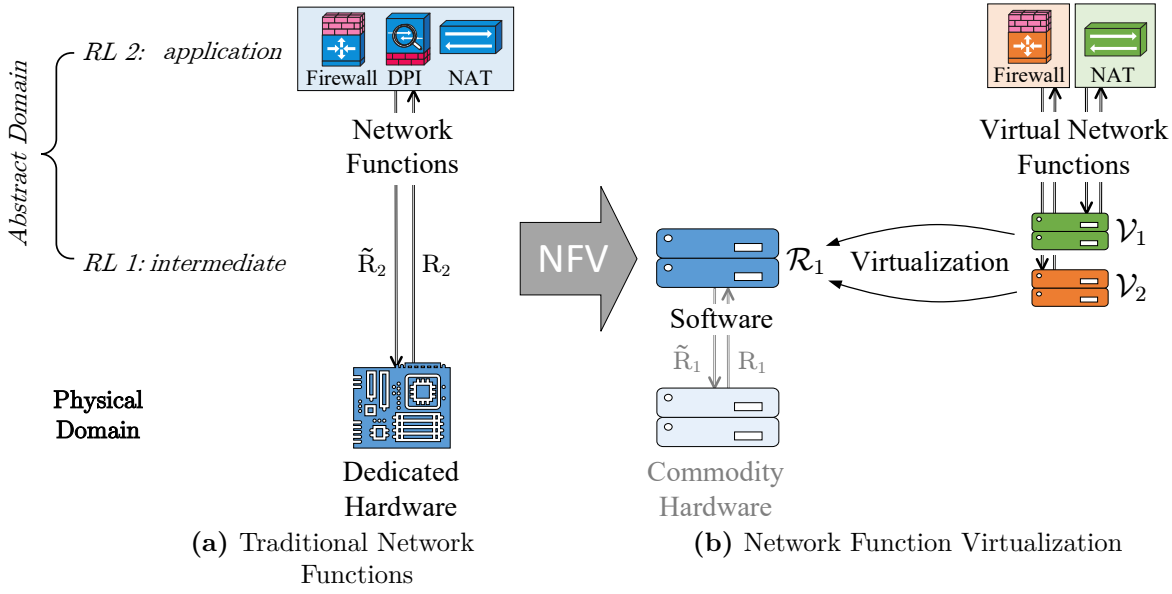


Figure 4.8: Network function virtualization uses the addition of a software representation level to subsequently run each network function (such as Firewalls, deep packet inspection (DPI), and network address translation (NAT)) in a virtualized environment. In this way, virtual network functions can quickly be reconfigured and managed more effectively. Network functions courtesy of [77]

In the context of future networks, and specifically future wireless networks, NFV is particularly useful as parts of the mobile core network such as the mobility management entity, the home subscriber server, and many other functions could be virtualized to enable flexible and dynamic operation. NFV represents a significant step towards network virtualization, as techniques from NFV could be used in network virtualization, although there are still significant challenges to overcome [77].

4.4.2 Wireless Network Virtualization

In this work, we distinguish between Wireless Link Virtualization (WLV) and Wireless Network Virtualization (WNV), which are often intertwined in the literature. Wireless network virtualization has been proposed as an extension of network virtualization to wireless networks, providing similar potential benefits in terms of flexibility and efficiency [78], [79]. Specifically, it is hoped that WNV can increase resource efficiency and flexibility for the problems of 1) the continuing rapid growth of demand for wireless services, 2) the ever-greater demand for diversity of services, and 3) the increasing costs of wireless infrastructure. Figure 4.9 shows the difference between wireless link virtualization and wireless network virtualization.

Similarly to network virtualization, we can think of WNV as using both node and

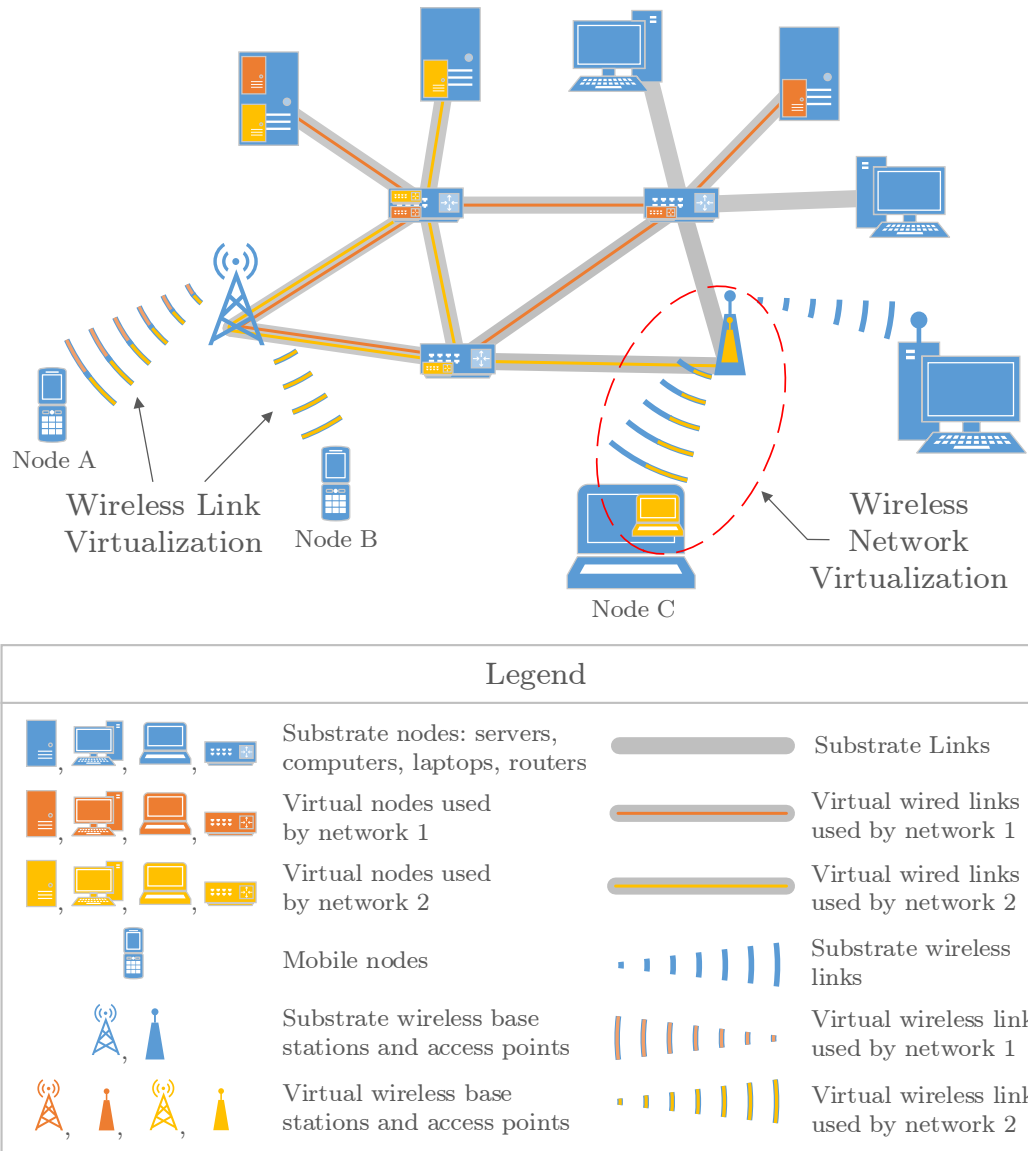


Figure 4.9: Wireless link virtualization and wireless network virtualization: Networks 1 (orange) and 2 (yellow) use network virtualization to coexist simultaneously on the wired network substrate. Wireless link virtualization is used to offer secure and customised services to nodes A and B, and these services can focus on different aspects such as high-throughput, low-latency, or low-power consumption, etc. In this example, node A has access to two types of services, one from networks 1 and 2 each, while node B only has access to the service from network 2. Wireless network virtualization is used to extend virtual network 2 to node C. Similarly to the previous figure, we do not show RF front-end virtualization, but imply it is used.

wireless link virtualization to create virtual wireless networks. Only when it is possible to virtualize both the node and link resources that make up the network, is it possible to consider (wireless) network virtualization [63] [80]. As we have seen previously, node virtualization and wired link virtualization are long-established techniques. However, wireless link virtualization is still at an early stage of development, and without wireless link virtualization it is not possible to do wireless network virtualization. In fact, most

work to date that self-identifies as wireless network virtualization has actually been concerned with wireless link virtualization, since WLV is a prerequisite of WNV.

Although we are ultimately interested in wireless network virtualization, this thesis focuses on wireless link virtualization for several reasons:

1. WLV is a prerequisite for WNV, and the other prerequisite, node virtualization, can already be implemented through many existing techniques.
2. The main difference between NV and WNV is the nature of the links, and thus once link virtualization has been accomplished, we could reuse NV techniques for other aspects of WNV, such as coordinated node and link mapping [81].
3. To date, work on WNV so far has almost exclusively focused on WLV, although it has self-identified as WNV. However, there are not yet many solutions for WLV, and the existing solutions also have some limitations, as will be shown in the next Chapter.

Software Defined Radio

One similarity between wired and wireless network virtualization is the requirement of being able to use resources in the abstract domain, before they can be virtualized. Just

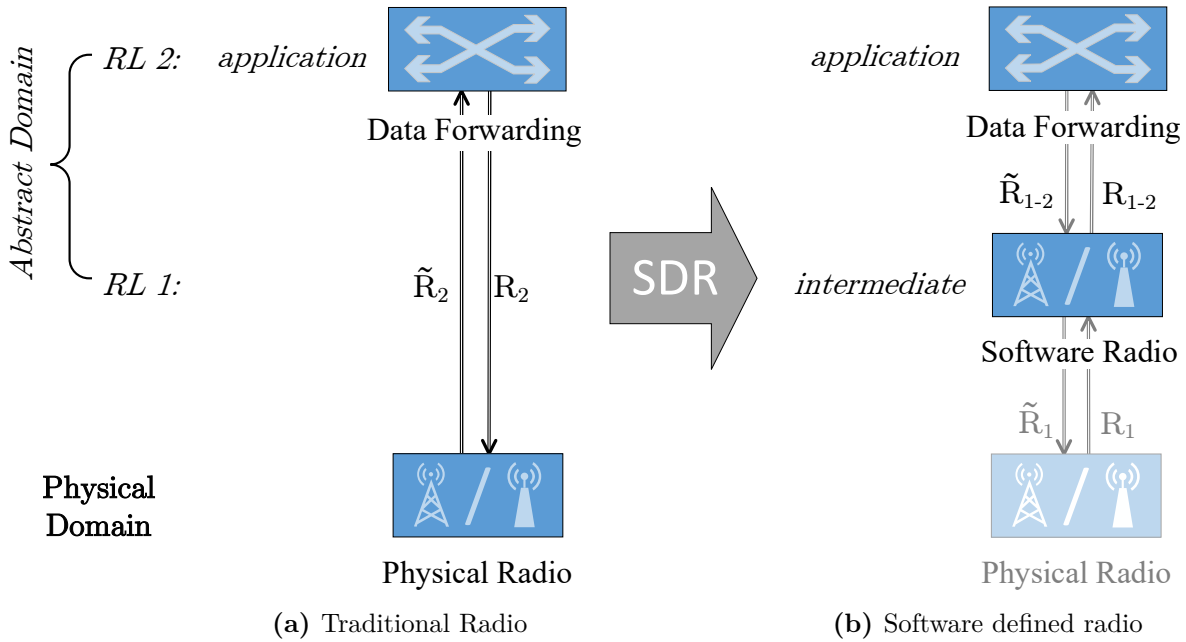


Figure 4.10: SDR is a prerequisite for radio resource virtualization, as it moves the compute cycle to the abstract domain.

as in wired networks, SDN has been employed to provide flexibility and dynamicity, in wireless networks SDR has been developed for similar reasons. SDR aims to take advantage of the flexibility and reconfigurability of software to perform the Digital Signal Processing (DSP) required for radio transmission and reception in a dynamic manner. In the same manner that SDN adds an additional representation level, SDR also uses a software representation level to enable reconfigurability, as shown in Figure 4.10. Software defined radios will be discussed in more detail in Section 6.4 in Chapter 6.

4.5 Summary

In this chapter, we examined the virtualization of different types of resources in computer networks. We reviewed the different node functions and how each function can be virtualized. Next, we described the virtualization of wired and wireless links – an open research problem in the wireless case. Lastly, we examined wired and wireless network virtualization, and conclude that the biggest obstacle to wireless network virtualization is the lack of wireless link virtualization. Therefore in the next chapter we survey existing WLV techniques (although many of these techniques self-identify as wireless network virtualization) to identify open research problems.

5 Survey of Wireless Link Virtualization

5.1 Introduction

Now that we have a clearer understanding of virtualization, in this chapter we survey the existing literature on Wireless Link Virtualization (WLV). Aided by the theory of virtualization, we provide the second major contribution of the thesis; a systematic examination of wireless link virtualization, leading to new insights, which, to the best of our knowledge, have not been described elsewhere. Many open research problems appear from these insights, and we believe that addressing these research problems is vital for WLV.

5.2 Test for Virtualization

The first step in performing the survey is to create a test for virtualization, to decide whether a technique that self-identifies as virtualization actually is virtualization. We return to the definition of virtualization developed in Chapter 3 to define such a test.

According to the definition of virtualization in 3.6, virtualization is always recursive in theory. Therefore, one possible test for virtualization is to check for recursion. We propose a test which determines whether proposed techniques are recursive, at least in theory.

The papers which we examine define abstract wireless link resources in some manner, and subsequently map these resources to different users based on some criteria. The allocation of resources to users is done through the use of an algorithm, whether this is an optimization, a heuristic method, an auction-based approach, or any other resource

allocation method. These resource allocation methods all have one thing in common; they have a set of **input** resources and a set of **output** resources. Input resources are the wireless link resources that algorithms allocate, while output resources are the resources that users receive from the algorithms. In general, input resources are relatively straightforward to determine, but output resources can be harder to identify.

Therefore we base our test for recursion on the mapping of input to output resources. We define recursive techniques as techniques in which no difference exists between input and output resources. In other words, the *types* of both resources are the same, and both the input and the output resources can be thought of as at the *same* representation level. Techniques for which this is true can be considered as recursive, and hence as virtualization, and thus we can call the input and output resources real and virtual resources respectively.

The recursive test for virtualization is very simple:

Recursive Test for Virtualization
1. Check the units of the input resources. 2. Check the units of the output resources. 3. If the units are the same, then the technique can be considered recursive and is virtualization.

Therefore, techniques for which the input and output resources types differ cannot be considered recursive. In this case the input and output resources are at different representation levels. Users cannot use the same technique again to split or aggregate resources. These techniques can be considered methods of resources sharing/combination, but not virtualization.

For example, a technique for which the input is spectrum resources (measured in Hertz) and the output given to users is also spectrum resources (also measured in Hz), is considered a recursive technique, since users could use the same technique again to virtualize their own spectrum resources, and split or aggregate the resources in a new way.

However, a technique which has as input spectrum resources (measured in Hz), and as output information rate (measured in bit/s) can not be considered recursive, since the units of the input and output resources are different. Users could not reuse this technique to split or aggregate their information rate.

5.3 Survey Overview

The survey analyses over sixty works from 2003 onwards on wireless network virtualization. However as discussed already, these works are in fact concerned with wireless link virtualization.

We apply the recursive test to works which self-identify as virtualization, to determine whether these works are also deemed virtualization according to the formal definition of virtualization developed in Chapter 3.

We consider works on any type of wireless link technology, and at any layer of the Open Systems Interconnection (OSI) communication stack. Only papers that propose resource mapping techniques are included, and thus many framework and architecture papers are omitted.

We split the papers into two groups: those that fail the recursive test, described in Section 5.4, and those which pass the recursive test and therefore deemed to be relevant, analysed in Section 5.5.

5.4 Non-Recursive Techniques

We briefly discuss the papers that did not pass the test, and why they are considered resource allocation techniques rather than virtualization techniques, according to our strict definition of virtualization. These papers are shown in Table 5.1. Although these works are not virtualization according to our definition, these techniques still represent significant contributions in wireless link resource allocation and can share (or sometimes, combine) wireless link resources in an efficient and fair manner.

The works in Table 5.1 are organised by the type of resources that they allocate and by date. We found several types of papers which consider spectrum resources as input, whether isolated in frequency, frequency and time, or space and other dimensions.

The techniques discussed in this section provide clever and efficient methods to share limited spectrum resources efficiently, they take as input spectrum resources, but the output resources that users receive are at the information representation level (see next section). Although this leads to better spectral efficiency and more optimal use of resources, this means that these techniques cannot be considered virtualization.

Table 5.1: Wireless resource allocation techniques

First Author	Reference	Input Resource (Units)	Output Resource (Units)	Recursive?
Lu	[82]	Subchannels (KHz)	Rate (Kbit/s)	No
Lv	[83]	Channels (MHz)	Bandwidth – i.e. Rate (Mbit/s)	No
Xu	[84], [85]	Subchannels (MHz)	Sum Rate (bit/s/Hz)	No
G. Liu	[86], [87]	Subchannels (MHz)	Rate (Mbit/s)	No
Kamel	[88]	Subchannel (MHz)	Rate (Mbit/s)	No
G. Zhang	[89], [90]	Subcarrier (MHz)	Data Rate (bit/s)	No
Feng	[93] - Spectrum	Spectrum (MHz)	Traffic (Mbit/s)	No
Cai	[94]	Radio Resource (KHz)	Utility (Bit/s/\$)	No
Chen	[95]	Channel (MHz)	Utility (Profit/bitrate)?	No
Liang	[96]	Radio Resource (MHz)	Utility (?)	No
Khatibi	[91]	Radio Resource Unit (MHz)?	Rate(Mbit/s)	No
Rahman	[92] - LVN	Spectrum (MHz)	Rate (Mbit/s)	No
Fu	[97], [98]	Fraction of time of subchannel (ms, KHz)	Information Rate (Kbit/s)	No
Kamel	[99]	Physical Resource Blocks (KHz, ms)	Rate (Mbit/s)	No
Zaki	[100], [101] - Dynamic	Physical Resource Blocks (KHz, ms)	Load (Kbit/s)	No
Zhao	[102]	Physical Resource Blocks (KHz, ms)	Rate, Delay (Mbit/s, ms)	No
Li	[103]	Physical Resource Blocks (KHz, ms)	Date Rate (Kbit/s)	No
B. Liu	[104]	Physical Resource Blocks (KHz, ms)	Traffic Rate (Kbit/s)	No
Panchal	[105]	Physical Resource Blocks (KHz, ms)	Load (Kbit/s)	No
Kalil	[106] [107]	Resource Blocks (KHz, ms)	Throughput (Mbit/s)	No
Moubayed	[108]	Resource Blocks (KHz, ms)	Sum Rate (Kbit/s)	No
Kokku	[109], [110]	OFDM Blocks/Slots (MHz, ms)	Rate/QoS (Mbit/s)	No
Costa-Perez	[111]	LTE Resource Blocks (MHz, ms)	Rate (Kbit/s)	No
Wei	[112]	Physical Resource Blocks (KHz, ms)	Utility (bit/Joule)	No
Wang	[113]	Physical Resource Blocks (MHz, ms)	Throughput(Mbit/s)	No
Di Stasi	[114]	Channels in Space (Hz, Grid Location)	Throughput (Mbit/s)	No
Q. Zhu	[115], [116]	Subchannels and Power (Hz, Watts)	Utility (Hz/m ³)?	No
Fan	[117]	Bandwidth and Power (MHz, dBm)	Rate (Mbit/s)	No
K. Zhu	[118]	Subchannels, Power and Antennas (Hz, Watts, # of antennas)	Rate (bits/s)	No
Ahmadi	[119]	Spectrum, Antennas (KHz, # of Antennas)	Rate (Mbit/s)	No
Ahmadi	[120]	Spatial Stream (MHz, time block)	Rate (Mbit/s)	No
Rahman	[92] - CVN/RVN	Spectrum, Remote Radio Heads (MHz, # of)	Rate (Mbit/s)	No

Resource Allocation Techniques

5.4.1 Frequency-based isolation

As an example of the recursive test, we look at the works [82–92]. In these works, input spectrum resources are allocated as subchannels (in units of Hz) to users based on the rate (in bit/s) that can be achieved by users. Thus the output users are receiving is a data rate, rather than a number of subchannels. Although these methods advance the field of spectrum resource allocation, because the input and the output resources are not of the same type, these techniques cannot be considered recursive, and subsequently do not fall under our strict definition of virtualization. Similarly, the work [93] proposes a spectrum resource sharing technique that relies on estimating the users' traffic requirements, and thus the output resources are at a higher level of representation than the input resources.

5.4.2 Frequency- and Time-based isolation

There are also techniques which allocate input spectrum resources using frequency and time isolation, either based on users' estimated traffic loads and requirements, or based on the best rates that users can achieve. The works [100–106] can be considered in the first category, and take into consideration users' traffic requirements when assigning spectrum blocks. The second category contains the works [97–99]. Similarly, the works [109], [110], and [111] implement resource allocation schemes that allocate wireless resources based on the data rate requested by the virtual networks. These implementations are very impressive resource sharing techniques, but again because the techniques are not recursive, they cannot be considered virtualization.

5.4.3 Space-based isolation

Several works such as [114–120] propose allocating virtual spectrum resources using space-based isolation methods such as power control, and Multiple Input Multiple Output (MIMO) schemes. However, these schemes rely on the Channel State Information (CSI) to allocate virtual spectrum resources. Since CSI is an information-level resource, these schemes cannot be considered recursive.

Moreover space-based isolation can only be achieved if the resource users do not care which spatial resources they receive - in other words if the location of the users does not matter. The reason for this is that resources that are requested and allocated in a particular direction are different to resources that have no spacial restrictions. For example, allocating bandwidth to a spatial stream means that units of the resource are

now Hz in a particular direction of a coordinate system, and we can think of the spectrum resources as changing from a scalar to a vector resource.

In fact, directed space-based isolation schemes enable spatial densification, and MIMO techniques can be seen as analogous to increasing the number of wireless nodes [121]. Taking this perspective clarifies why directed spectrum resources are not the same as virtual spectrum resources.

5.5 Recursive Techniques

The papers which pass the recursive test are shown in Table 5.2, where we have combined the input and the output resources types for brevity (since they are the same). We discovered that the recursive test could also be used to determine the representation level of a virtualization technique. Methods which have the same unit types (for example, Hz, KHz, MHz, etc.) can be grouped together at the same representation level. We found that there were two representation levels for wireless link resources:

1. Spectrum Level Virtualization, which is the virtualization of wireless resources in the form of spectrum, such as [122].
2. Information Level Virtualization, also known as data path or flow-level virtualization, which refers to the virtualization of the data carried by spectrum, and allows wireless links to be shared or combined at the information level, such as [78].

The most important consideration for virtualization is the representation level at which virtualization takes place. Therefore, in table 5.2, we first classify papers by representation level. Section 5.6 discusses spectrum-level virtualization techniques, while Section 5.7 discusses information-level techniques.

The next consideration is the form of isolation used at a representation level, for example frequency, time, or space isolation. We examine the isolation methods used in the literature, and in particular the isolation methods that seemed to be more successful. Therefore in the table, the second classification is by isolation dimension(s), ordered by date.

A third consideration of the survey is the embedding method used by virtualization techniques. Due to space limitations, the different embedding methods are described in Table 5.3. A final aspect that we examined was the type of mapping that techniques performed, which was very heavily focussed on many-to-one mapping, i.e. partitioning.

5.6 Spectrum Level Virtualization

To the best of our knowledge, spectrum-level virtualization was first proposed in the work [123], in which wireless network virtualization is considered as an extension to the wired network virtualization testbed GENI. The authors of [123] conceptually discuss several methods of isolation at the spectrum level, such as along the frequency, time, space, and code dimensions or a combination of these. The authors of [21] propose a virtualization framework for wireless networks and discuss some of the steps involved in deploying virtual wireless networks.

5.6.1 Frequency-based Isolation

One of the first proposals of a spectrum virtualization technique was proposed by Forde *et. al* [16]. This work proposed an auction-based method for mapping virtual spectrum bands to real spectrum bands, and simulated the allocation of resources to maximize the revenue for the auctioneer.

The work [124] implements the Spectrum Virtualization Layer (SVL) that offers virtual spectrum resources to users according to their desired bandwidth. In this way the limitations of real spectrum with fixed bandwidths and non-contiguous segments can be overcome. The virtual spectrum is mapped to real spectrum according to a pre-defined spectrum map. The spectrum mapping can map the virtual baseband to a real spectrum band with the same bandwidth, a narrower bandwidth, or to several non-contiguous bands with a greater combined bandwidth. However, SVL is many-to-one in terms of wireless link virtualization, as it cannot combine several real links into one virtual link. The authors show that SVL can be used to create a virtual WiFi Network over TV White Space channels, and that SVL could be used to combine the access points of several different technology standards.

The authors of [122] and [125] propose a channel virtualization scheme to share spectrum opportunistically, based on the price that virtual networks are willing to pay for virtual channels. The requirements of virtual networks are split into baseline and fluctuant parts. In this work the virtual channels used for the fluctuant requirements can be shared between different virtual networks and thus there is a probability of collision. The authors propose several algorithms to assign the virtual resources in such a way that maximizes the revenue for the owner of the real spectrum, and perform simulations to compare these and other algorithms from wired network virtualization.

Table 5.2: Wireless link virtualization techniques: Isolation

	First Author	Reference	Input & Output Resource Type (Units)	Recursive?	Representation Level	Isolation	Mapping
Virtualization	Forde	[16]	Bands (MHz)	Yes	Spectrum	Frequency	$m \mapsto 1$
	Tan	[124]	Baseband (MHz)	Yes			$m \mapsto 1$
	Yang	[122], [125]	Channel (MHz)	Yes			$m \mapsto 1$
	Zaki	[100], [101] - Static	Physical Resource Blocks (KHz, ms)	Yes		Frequency and Time	$m \mapsto 1$
	Yang	[126]	Physical Resource Blocks (KHz, ms)	Yes			$m \mapsto 1$
	van de Belt	[127], [128]	Physical Resource Blocks (KHz, ms)	Yes			$m \mapsto 1$
	Hsu	[129]	Bandwidth and Time units (KHz, ms)	Yes			$m \mapsto m$
	Lu	[130]	Physical Resource Blocks (KHz, ms)	Yes			$m \mapsto 1$
	X Zhang	[131]	Timeslot and (x, y) coordinates in Space (ms, (meter, meter) from [132])	Yes		Time and Space - Power	$m \mapsto 1$
	Chandra	[78]	Information Throughput (Kbit/s)	Yes	Information	Time - Switching	$m \mapsto 1$
	Smith	[133]	Throughput (Kbit/s)	Yes			$m \mapsto 1$
	Coskun	[134]	Throughput (Kbit/s)	Yes			$m \mapsto 1$
	AlHazmi	[135]	Packets (Bytes)	Yes			$m \mapsto 1$
	Lv	[136]	Packets (Bytes)	Yes			$m \mapsto 1$
	Mahindra	[137]	Throughput (Mbit/s)	Yes		Time - Addressing	$m \mapsto 1$
	Perez	[138]	Packets (Bytes)	Yes			$m \mapsto 1$
	Sherwood	[139]	Throughput (MBit/s)	Yes			$m \mapsto 1$
	Bhanage	[140], [141]	Throughput (Mbit/s)	Yes			$m \mapsto 1$
	Xia	[142]	Throughput (Kbit/s)	Yes			$m \mapsto 1$
	Aljabari	[143]	Throughput (Kbit/s)	Yes			$m \mapsto 1$
	Nakauchi	[144]	Throughput (Mbit/s)	Yes			$m \mapsto 1$
	Katsalis	[145]	Throughput (Mbit/s)	Yes			$m \mapsto 1$
	Feng	[93] - Flow	Rate(Mbit/s)	Yes			$m \mapsto 1$
	Mahindra	[137]	Throughput (Mbit/s)	Yes		Space - Power	$m \mapsto 1$

5.6.2 Frequency and Time based Isolation

The works [100] and [101] first propose a spectrum virtualization method involving frequency and time isolation. The idea in these works is to virtualize Long Term Evolution (LTE) base stations, also called evolved Node Bs (eNodeBs), using a machine hypervisor that also has control of the spectrum resources. The virtual eNodeBs can request Physical Resource Blocks (PRBs) from the hypervisor. The authors propose two different algorithms - a static algorithm where the virtual PRBs are assigned once, and a dynamic algorithm which depends on the operator's load. However, only the static algorithm can be considered virtualization, because the dynamic algorithm fails the test for recursion since the hypervisor assigns virtual PRBs based on a higher level of representation (information-level metrics such as channel conditions, user loads and Quality of Service (QoS) requirements).

The authors of [126] propose a virtual resource embedding algorithm for frequency and time domain resources. In this work, a number of contiguous virtual frequency and time-domain resources can be requested for a duration of time periods. Virtual resource requests can be chosen with several priority levels, corresponding to the likelihood of acceptance of a request. The heuristic algorithm proposed is based on finding the most suitable location for a request using Karnaugh-map theory and the embedding density, a measure of how efficiently resources are used. The authors compare several algorithms through simulations in terms of resource utility and the acceptance ratio of virtual resource requests.

Our work [127], which will be discussed later in the thesis in Chapter 8, extends the Karnaugh-map algorithm to the dynamic case, which allows virtual resources to be re-embedded at every time period. Through simulations, we show that this uses resources in a more efficient manner and leads to a higher request acceptance ratio. We also propose a greedy dynamic embedding algorithm which has increased performance compared to the dynamic Karnaugh-map algorithm, and formulate the objective problem for the virtual resource embedding problem with prioritised requests.

In another work, [128], also discussed later in the thesis, we demonstrate how virtualization could be applied to a use-case scenario, in the context of a virtual public safety operator. We show that virtualization enables spectrum resources to be used in a flexible manner that guarantees connectivity for public safety users during an emergency, but allows spectrum resources to be used by commercial operators as needed during day-to-day operation.

The work [129] proposes a dynamic virtual resource embedding algorithm that allows

virtual resource requests to be fulfilled through multiple virtual resource blocks. In other words the real resource owner can aggregate spectrum blocks to fulfil resource requests. This use of spectrum aggregation means that the embedding used is potentially many-to-many.

In the work [130], the dynamic greedy algorithm developed in [127] is extended by using a genetic algorithm to find the most fit embedding location for a virtual resource request. The authors show that the genetic algorithm offers an improvement over the dynamic greedy algorithm in terms of resource utility and rejection rate.

5.6.3 Time and Space based Isolation

The work [131] proposes a time and space based isolation method to allocate wireless network resources to experiments on a grid. The objective is to fit as many experiments into a limited grid, while maintaining strict isolation. The allocation takes into account the distance between nodes as a means of modelling potential interference. The authors show that using space-based isolation in addition to time-based isolation can provide significant gains in resource utilization. Gains are more significant for small scale experiments than for larger experiments. One important aspect to note is that space-based isolation is possible in this case because the location of the nodes does not matter. However, in more commercial applications this assumption most likely will not be true, and therefore space-based isolation will not be possible (see Section 5.4.3).

5.6.4 Discussion

Spectrum-level virtualization has so far mainly focused on frequency isolation, and the combination of isolation by frequency and other dimensions such as time. This could be because frequency isolation is already extensively used in networks as frequency division multiplexing.

There are several reasons that could explain why frequency isolation is so widely used. The first is that due to the nature of spectrum resources, different parts of the spectrum are suitable for different applications and thus it is intuitive to divide up the spectrum into bands for different applications. Another reason why frequency isolation could be preferable to time isolation is that it allows wireless links to be operational at all times, whereas using time isolation this is not possible. Time isolation also requires that nodes in a network be synchronised tightly.

Table 5.3: Wireless link virtualization techniques: Embedding

	First Author	Reference	Isolation	Embedding
Spectrum Level	Forde	[16]	Frequency	Auctioning of paired spectrum bands based on price and distance between bands
	Tan	[124]		Implementation of spectrum virtualization based on predefined spectrum map(s)
	Yang	[122], [125]		Baseline and varying channels with collision probability. Comparison of heuristic, genetic, and optimal allocation algorithms based on price
	Zaki	[100], [101] - Static	Frequency and Time	Predefined spectrum allocation
	Yang	[126]		Heuristic embedding based on Karnaugh-map areas
	van de Belt	[127], [128]		Dynamic greedy embedding and dynamic version of [126]
	Hsu	[129]		Bottom-Left algorithm with splitting of resources
	Lu	[130]		Dynamic genetic algorithm for Karnaugh-map areas
	X Zhang	[131]	Time and Space - Power	Embedding based on distance between nodes and experiment duration
Information Level	Chandra	[78]	Time - Switching	Time-switching at either fixed intervals or adaptive intervals based on traffic load
	Smith	[133]		Synchronised time-switching in a round-robin fashion
	Coskun	[134]		Fast time-switching at regular intervals using PSM and PCF features
	AlHazmi	[135]		Extension of [134] with more detailed analysis of virtualization capabilities
	Lv	[136]		Round-robin time-switching with opportunistic rebroadcasts
	Mahindra	[137]	Time - Addressing	Policy manager limits each experiment to a predefined maximum rate
	Perez	[138]		Weighted round-robin scheduler
	Sherwood	[139]		Priority resource allocation policy configured by network administrator
	Bhanage	[140]		Based on airtime quotas allocated at hardware setup.
	Bhanage	[141]		Weighted sharing based on an airtime fairness metric
	Xia	[142]		Not clear how vMAC interfaces are embedded but possibly through rate control
	Aljabari	[143]		No specific embedding - limited by machine performance
	Nakauchi	[144]		Embedding based on a weighted fraction of airtime
	Katsalis	[145]		Allocated according to predefined throughput share guarantees
	Feng	[93] - Flow		Flow scheduler allocates flows to virtual networks based on traffic load
	Mahindra	[137]	Space - Power	Policy manager limits each experiment to a predefined maximum rate

Space isolation is also used widely in wireless networks and allows the spectrum to be reused in multiple locations. However space isolation is possibly the most difficult form of ensuring independence between different links, due the unreliable and stochastic nature of propagation of radio waves.

With the exception of [124], to date the work on spectrum virtualization has focussed heavily on simulation and on efficient embedding schemes, rather than on implementing virtualization techniques with strict isolation. One of the reasons for this might be that most of the works deal with cellular technologies, and thus there is an emphasis on efficiency, as it is costly and time-consuming to add additional capacity to these types of networks. Another reason could be the challenges of implementation of virtualization in these networks, since many techniques for cellular networks rely on strict synchronisation.

5.7 Information Level Virtualization

The concept of wireless network virtualization was first seen as an extension to wired network virtualization [123], [146], and initial wireless link virtualization techniques focused on information-level link virtualization, which had already been accomplished for wired networks (see section 4.3.1). The idea of Virtual Local Area Networks (VLANs) had also been established for wired networks, and so some of the first wireless link virtualization papers focused on finding methods to allow multiple virtual links to coexist on the same physical wireless device at the information level.

5.7.1 Time-based Isolation using Switching

One of the first papers to describe and implement a method of wireless link virtualization was [78]. In this paper, the authors address the problem of connecting a wireless node to multiple links simultaneously, which up until this point could only be done using multiple wireless front-ends. To solve the problem of excessive battery consumption and the expense of multiple wireless cards amongst other reasons, the MultiNet solution was developed, which time-switches a wireless card between multiple virtual links at regular intervals, or at adaptive intervals based on traffic load. Virtual interfaces are presented to the node at the information level (Media Access Control (MAC) layer), and the node can send and receive packets through one or more interfaces simultaneously. The interfaces appear as if they are constantly active, but in practice the device driver buffers packets until the right virtual link is active. This solution reduced the energy consumed dramatically compared to using multiple cards, since two radios consumed

about double the energy consumed by MultiNet. However, the process of switching between links means that the association procedure is initiated every time a switch occurs, and leads to a decrease in performance in terms of delay, throughput and packet loss when compared to solutions involving multiple wireless cards.

The work [133] proposes a round-robin time-switching approach for virtualizing wireless links used by virtual nodes. These virtual links are co-ordinated centrally to ensure synchronization between nodes. However, because of propagation delay, some overhead is incurred when switching from one virtual link to another and this leads to a relatively high packet loss and increased latency.

To deal with the issues of long handover times when switching between virtual links, the authors of [134] use the Power Saving Mode (PSM) to shorten delays compared to virtualization without PSM enabled. They also take advantage of the Point Coordination Function (PCF) features of wireless cards to avoid repeating the association procedure for every switch. In [135], an extension of [134], the authors investigate the effects of virtualization on traffic properties.

The work [136] examines time-switching in the context of wireless mesh networks, and proposes a round-robin scheme to broadcast packets from differing virtual networks. Because links can be unreliable, and no acknowledgement is used when using the broadcast mode, the algorithm rebroadcasts certain packets based on a successful reception probability.

However some of the disadvantages of the time-switching approach are that a high level of synchronization is required between different nodes, and also that when there are many virtual links the delay for each link increases, which affects the performance of delay-sensitive applications.

5.7.2 Time-based Isolation using Addressing

The concept of Virtual Access Points (VAPs) was first introduced in [147] as a means of allowing multiple access points (and thus multiple links) to coexist simultaneously on one physical access point. Each VAP is allocated a unique Service Set Identifier and capability set. The use of addressing in this way provides time isolation at the information level (MAC layer) and enables VAPs to be indistinguishable from physical access points [147]. VAPs cannot emulate the operation of a physical Access Point at the radio frequency layer (Spectrum level) unless multiple physical radios are available. This limitation requires all virtual links to use the same physical radio parameters such as channel. However,

VAPs can differentiate at the information level, offering different throughput rates, packet loss, and latency, etc.

One of the first papers to implement Virtual Access Points is [137]. This paper compares space-based and time-based isolation for virtualization on the ORBIT testbed. Space-based isolation will be discussed later. For the time-based isolation the authors prefer the VAP approach rather than the time-switching approach, because of the disadvantages of the time-switching approach already discussed. A policy manager limits the maximum rate that each experiment can achieve. The authors show that using VAPs adds minimal overhead to the throughput performance and the delay compared to the conventional approach.

The work [138] investigates a virtualization approach that stores packets from different virtual operators in separate queues until the scheduler sends packets to the wireless interface using a weighted round-robin scheme. Although this approach does not use VAPs, it also uses packet addresses to isolate at the information level.

Similarly, in [139], addressing-based isolation is used to partition the network into different ‘slices’, which can have different data rates set by the network administrator. Address-based isolation is also used in [141] to create a virtual Worldwide Interoperability for Microwave Access (WiMAX) base-station that can offer slice customization to its users. The resource allocation is based on airtime quotas configured at setup.

A method for controlling the air-time usage of different virtual links is proposed in [140] and [144]. The virtual links are isolated using virtual access points and the slice identifiers are used to monitor and control air-time usage of each link.

Virtual WiFi [142] proposes a method of having multiple virtual MAC (vMAC) interfaces for virtual machines on the same physical wireless device. This is achieved through giving a different address to each vMAC interface and assigning each virtual machine a MAC address. However, it is not clear whether vMACs have any resource limits (apart from the hard performance limits of the device).

The authors of [143] also use VAPs to deploy multiple wireless links on a shared physical infrastructure. The focus is on open-source virtualization of access points and wireless link interfaces. However, the embedding used is not described.

In [144] virtual links are isolated through MAC addressing, and each virtual link has its own queue with different parameters. The algorithm to assign wireless airtime to virtual links is based on a weighted fraction. The authors show that the desired fractions of airtime can be achieved quite closely using their resource allocation algorithm.

The CONTENT project aims to investigate end-to-end virtualization in heterogeneous wireless and optical networks. In the project, wireless virtualization is done by

assigning each flow a virtual identification tag, similar to the above approaches [145]. The embedding is performed according to predefined throughput guarantees.

The work [93] developed two algorithms, one a spectrum-level algorithm and the other a flow/information level technique that uses addressing to allocate virtual packets based on traffic load. However only the flow level technique can be considered virtualization, since the spectrum level technique allocates spectrum also based on traffic load, and thus is not recursive.

5.7.3 Space-based Isolation

As mentioned previously, the authors of [137] also examine space-based isolation for virtual links. Isolation is achieved between nodes on a shared channel by using sufficient spatial separation to avoid interference and by managing the power levels of transmissions. Crucially, this work is concerned with allocating experiments on a grid, and space-based isolation is appropriate in this scenario, since it does not matter *where* experiments are located in space.

5.7.4 Discussion

Although time-switching was initially found to be a potential solution for wireless link virtualization at the information level, it soon became apparent that the delays between switching led to significant performance decreases in terms of throughput and especially latency. Another problem with time-switching is the need for precise synchronization between different nodes. The addressing-based isolation method provides many advantages in this regard compared to time-switching.

However, the addressing-based solution could have some overheads compared to time-switching since every node receives all of the traffic, and packets are only filtered out at the receiving nodes. This could lead to reduced battery life, an important point to consider for wireless nodes.

The space-based approach does not seem to offer many advantages compared to the time-switching or the addressing-based approaches, since isolation between different virtual links is very difficult to achieve reliably, and this affects the performance of the virtual links.

In terms of the embedding techniques used, the majority of works employ either predefined resource allocation, or round-robin style allocation. Only [78] and [93] propose embedding schemes based on traffic load. It is clear that the focus of information level

techniques is on achieving isolation and actually implementing virtualization. A possible explanation of the emphasis on isolation is that these virtualization schemes are used in scenarios where the resource capacity can easily be increased, such as WiFi.

5.8 Comparison with Existing Surveys

As previously mentioned, there are several surveys concerning wireless link and network virtualization, amongst which the most important are [18, 19, 79, 148, 149]. We briefly compare the approaches taken by these works to the survey in this chapter, and highlight how our survey offers additional and complementary insights to the existing literature. Typically such a review would come before performing the survey; however, in this case, now that the survey is complete, it is easier to show how the approach taken in this thesis provides additional understanding.

5.8.1 Focus on Purpose / Technology

In general, existing surveys define virtualization in terms of its perceived purpose (e.g. in terms of business models, benefits etc.), and classify virtualization techniques by purpose and technology. For example in [149] wireless link virtualization techniques are grouped by access technology, such as IEEE 802.11 (WiFi), 3GPP LTE, IEEE 802.16 (WiMAX), while in [148], [79], and [19] techniques are grouped by technology and purpose – whether techniques have been developed for commercial or experimental use.

Although the above approaches are useful for giving a detailed description of work that has been done to date, it can be hard to identify universal virtualization trends and potential research topics. By taking a purposely technology-agnostic approach based on a strong definition of virtualization, in this thesis we are able to classify virtualization techniques theoretically. The advantage of this is that it is easier to identify common trends (described in the next section) and propose problematic research areas in a clear and coherent manner. In addition, the technology-agnostic approach has another benefit; it can still be relevant to future technologies when these are developed.

As a specific example, compare the above Table 5.2 with Table I found in [149]. Both tables cover a similar collection of papers (some papers found in [149] do not pass the recursive test, and are omitted), but in the latter table techniques are grouped by technology, whereas in this work techniques are grouped by representation level. In the technology-oriented table, all of the works on LTE ([100, 101, 126, 127]) can be

found at the spectrum-level in the other table, and the works on IEEE 802.11 ([133, 137, 142, 144]) can be found at the information-level. By focusing on technology, one might wrongly assume that virtualization in IEEE 802.11 can only be done through information-level techniques such as traffic shaping, or that virtualization in LTE can only be achieved by spectrum virtualization. However, virtualization can always be done at any representation level; for example, 802.11 virtualization occurs at the spectrum-level in [124]. The *observation* that to date most work in LTE has been at spectrum-level, and most work in WiFi has been at information-level is only possible because of the technology-agnostic approach.

Thus the focus on the purpose/technology of virtualization techniques offers certain insights and enables the detailed description of virtualization techniques, but it can mean that general trends and new possibilities are missed.

5.8.2 Lack of Specificity

The theoretical approach underpinning this survey provides a second advantage over existing surveys; it offers a specific language that is clear and succinct, which can lead to new insights about virtualization.

Several of the concepts described in this thesis are briefly mentioned in the current literature. For example, the need for isolation in virtualization is mentioned in almost all surveys listed above. However, although isolation is recognised as important, it is not explicitly described. In [19] and [149] isolation is used for the survey tables, although a vast range of different isolation measures are given, among which are: MAC layer, time division/slot, PRBs, packet, spatial, flow, traffic shaping, rate control, slice, sub-channel/sub-carrier. The large number of isolation types does not provide clarity, but instead means that isolation is left vague. By using a theoretical approach in this survey, a clear understanding of isolation is reached, and we can identify the specific isolation types such as frequency, time, etc., even when different terms are used.

Similarly, several works have also touched on the concept of ‘levels of virtualization’ or ‘depth of virtualization’, such as [18–20, 80, 109, 148, 150]. Again, this concept is insufficiently described and is unclear. Often ‘levels’ of virtualization are described as spectrum-, infrastructure-, and network-level virtualization. In fact, the first two terms are simply *link* virtualization and *node* virtualization respectively, and the third is the combination of both, i.e. network virtualization. These are not “levels” in any meaningful sense, just different applications of virtualization.

The term ‘level’ ([19], [149]) is also used to refer to different levels of the protocol

stack, or alternatively ‘depth of virtualization’/‘perspective’ ([148], [18]) is used. Some of the levels described are ‘flow-level virtualization’, ‘protocol-perspective’, and ‘front-end- and spectrum-perspective’. However, there is no clear distinction between any of these levels or method to recognise the virtualization level. In order to be specific and to avoid introducing ambiguity when discussing different levels of representation (i.e. virtualization levels in other works), we started this thesis with a theoretical overview of Abstraction/Representation (AR) theory in Chapter 2, which ultimately allows us to be specific when surveying the literature.

Finally, by providing specificity, we are able to derive a test for virtualization, which allows us to distinguish virtualization and non-virtualization techniques with some validity. Without specificity, the classification of works can seem arbitrary, as one does not know the difference between one class and the next.

5.9 Insights & Discussion

Like any survey, there are many insights that can be drawn from our systematic examination of the work to date on wireless link virtualization, such as 1) the need for performance metrics to measure the benefits of virtualization quantitatively, 2) the extreme lack of work on the use of virtualization to aggregate resources, 3) analysis on how users discover which resources are available, predict which resources they require, request available resources, and which request strategies to adopt. However, while the above insights can provide fruitful research topics to pursue, the most important structural insights are found when stepping back to examine the big picture.

Figure 5.1 summarizes the different aspects of wireless link virtualization visually, based on the theory developed previously in Chapter 3, and lists the possible options for each aspect of virtualization. When keeping this overview picture in mind, it is possible to observe significant insights, which we describe in subsections 5.9.1 – 5.9.5.

5.9.1 Use of Multiple Representation Levels

The first major insight is that many works attempt to virtualize resources through the use of multiple representation levels, listed in Table 5.1. However, for work to qualify as virtualization according to our theory of virtualization, it must always occur within one representation level in the abstract domain, as discussed in Chapter 3.

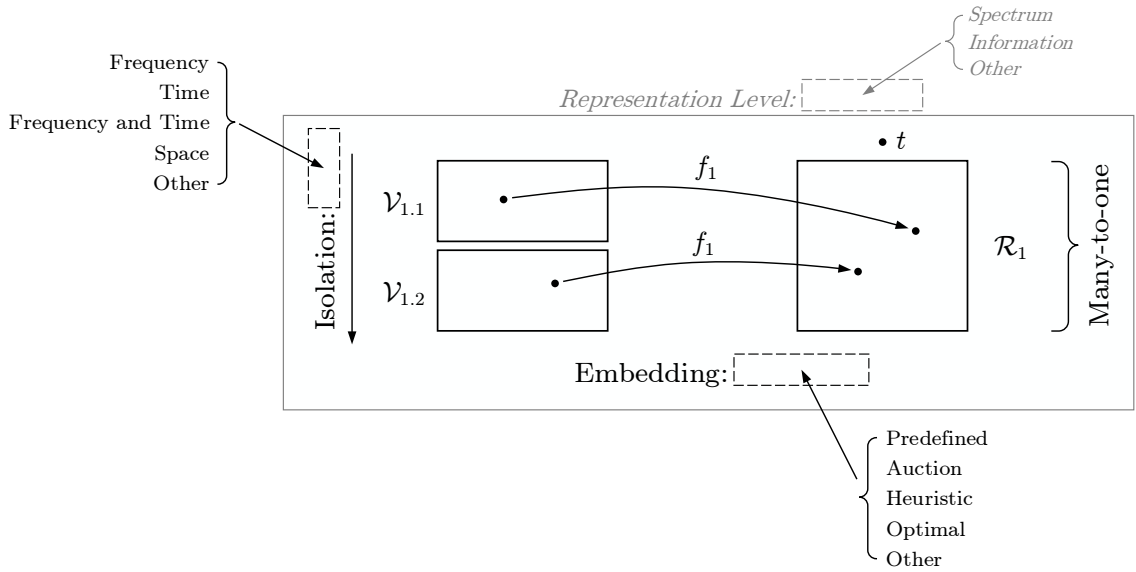


Figure 5.1: Summary of the most important aspects of wireless link virtualization; the representation level at which virtualization is done, the isolation methods used, and the embedding techniques employed. The most important insights from the survey are that 1) virtualization is often attempted across both the spectrum and information representation levels, which is not possible, 2) work is often exclusively focused on either the isolation or the embedding problem, 3) work is largely skewed towards the embedding problem, and 4) almost no work exists on the isolation problem at the spectrum-level.

We found that only approximately 41% of the surveyed works that self-identify as virtualization can actually be considered virtualization according to our strict definition of virtualization. Therefore, despite the valuable contributions which the other 59% of works have made, they are not considered as virtualization in this thesis. The need for virtualization to occur within one representation level is an important lesson to bear in mind for future work.

5.9.2 Exclusive Analysis of Either Isolation or Embedding

A second important insight is that the large majority of works focus exclusively on either the isolation problem or the embedding problem separately, and do not consider that virtualization consists of both problems. Although it is useful to separate these problems to tackle them individually, it is important to recognise that virtualization requires *both* isolation and embedding.

5.9.3 Lack of Focus on the Isolation Problem

Following from the above insight, the majority of existing work has investigated the embedding problem, mainly through theoretical analysis and simulation, while assuming that virtual resources are isolated. While these works undoubtedly make valuable contributions, it is not possible to develop virtualization practically without solving the isolation problem. In fact, since the embedding problem depends on the isolation problem, it might not even be possible to use certain embedding techniques in practice, due to issues arising from isolation. Therefore, more emphasis is needed on the isolation problem.

A further research question which has barely been addressed in the literature is the form of isolation that is most appropriate for different applications and scenarios. The only work we are aware that compares forms of isolation is [137]. From the survey, we found that works at the spectrum-level primarily provide isolation (theoretically) through the frequency dimension, and sometimes combine this with an additional dimension, while at the information-level the isolation is almost exclusively done in the time dimension. However, further analysis on isolation is needed.

5.9.4 Exclusive use of Many-to-one Embedding

Another insight which can be drawn from the survey, is that almost all works have focused on splitting real resources, i.e. many-to-one virtualization, – whether at the spectrum level or at the information level. Although further research is needed to investigate this surprising result, one possible explanation is that for many years wireless link resources were viewed as abundant and were over-provisioned, but that as wireless communications have increased exponentially over the past few decades, existing resources now need to be split to accommodate the additional devices [151].

However, in recent years, large amounts of new spectrum are becoming available in the millimeter wavelength range (mmWave) [152, 153]. It may be more important to consider the aggregation of resources when mmWave is considered, as some proposals see wireless resources in the mmWave spectrum band paired with wireless resources in other spectrum bands (either conventional or different mmWave bands) [152, 154]. The aim of using multiple frequency bands is to use the unconstrained mmWave bands for data when possible, and the lower frequency bands for control information or only when necessary for data [152]. Similar methods have also been proposed for aggregating 3GPP LTE and/or IEEE 802.11 (WiFi) bands to take advantage of less constrained spectrum bands and to give a higher throughput [155–157]. Therefore, it is important to investigate

one-to-many and many-to-many techniques, as these will likely have applications in the future.

5.9.5 Lack of Isolation at the Spectrum Level

There are a number of works which do examine the isolation problem, but with the exception of the work [124], these works have focussed on isolation at the information level. As a result, the virtualization of wireless links has been implemented at the information level, and is being used commercially with success [158], providing many benefits such as the of sharing wireless resources and access points [78], increased security [159], and new usage scenarios [134].

However, despite the existence of wireless link virtualization at the information level, the potential benefits and drawbacks of wireless link virtualization at the spectrum level have barely been studied practically, as to the best of our knowledge, the only implementation of a link virtualization scheme at the spectrum level is the Spectrum Virtualization Layer [124].

One important consideration to bear in mind is that due to the stochastic nature of the wireless channel, it is not possible to guarantee a specific data-rate, unlike a wired connection. This could mean that spectrum-level virtualization can provide many advantages over information-level virtualization, since it would allow virtual link resources to be used with much greater control, flexibility, and customizability. Spectrum-level virtualization would also allow spectrum resources to be used more efficiently than information-level virtualization, since wireless links can be better optimized to the channel conditions and application requirements. This is extremely significant, as the electromagnetic spectrum is considered a very expensive and valuable resource, but it remains artificially underused in many cases [160].

However, the isolation required between different virtual links at the spectrum level might not be as straightforward as for the information level, and might require additional overheads in terms of initialization and management. Without practical implementations, it is not possible to quantitatively discuss these topics.

5.10 Acting on the Insights

It is clear that there are still many research challenges and unresolved questions on the topics of wireless link virtualization and wireless network virtualization. However, the

research topic which stands out the most from our examination of WLTV is that the work on the spectrum level is severely lacking, and especially on the isolation problem.

Therefore, in the remainder of the thesis, we act on the insights gained from the application of the precise definition to the literature on WLTV. Our objective is to rebalance research so that all aspects of virtualization are investigated thoroughly. We address the major research gaps in the following manner:

1. Concentrating on the largest research gap, in Chapters 6 and 7 we focus on the isolation problem at the spectrum level, and investigate how the isolation of virtual wireless links can be implemented.
2. We bear in mind that virtualization consists of isolation and embedding, and as a result we examine the embedding problem at the spectrum level in Chapter 8. Since a significant body of work exists already on the embedding problem, our investigation focuses on dynamic embedding, which aims to make the embedding process more flexible and efficient.

Without first developing the theory of virtualization, it would not have been possible to systematically survey the literature on wireless link virtualization, and to arrive at these insights.

6 Spectrum Level Virtualization: The Isolation Problem Part 1

Designing an implementation

6.1 Introduction

As we saw in the previous chapter, almost all implementations of wireless link virtualization to date have been at the information level. Although information-level virtualization provides many benefits such as the sharing of wireless resources and access points [78], increased security [159], and new usage scenarios [134], the benefits and drawbacks of wireless link virtualization at the spectrum level have barely been studied. The works which do exist, such as the spectrum-level works listed in Table 5.2, have almost exclusively been theoretical. Without real-world implementation it is difficult to discover possible applications of wireless link virtualization at the spectrum level.

Therefore, noting the lack of (implementation-based) research on *spectrum-level* wireless link virtualization, the remainder of the thesis is concerned with this topic. We split spectrum-level virtualization into its constituent components, namely the isolation and the embedding problems (see Chapter 3), and focus on each problem individually. In this chapter and the next, we address the isolation problem, while Chapter 8 explores the embedding problem.

The isolation problem is fundamentally about maintaining independence between different virtual resources, and as such can only truly be explored through implementation. Essentially, if one is able to create virtual resources that are completely segregated, then the isolation problem is solved. However, depending on the application, there could be multiple methods of creating virtual resources, and this is one of the interesting questions that has not yet been analysed. In this thesis, we investigate frequency isolation as a

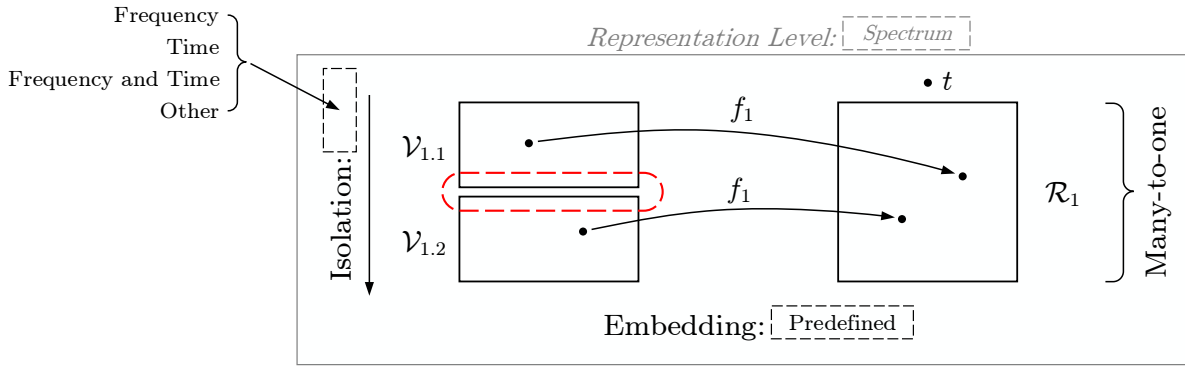


Figure 6.1: This example shows wireless resource virtualization at the spectrum level. The isolation problem is about maintaining independence between virtual resources, highlighted in red. Isolation can be done in different dimensions, such as the dimensions shown on the left of the figure. In this case, the embedding is predetermined, i.e. fixed allocation.

starting point, and hope to extend our implementation to frequency and time isolation in the near future. Figure 6.1 depicts the isolation problem visually.

Therefore, our aim is to solve the isolation problem for wireless link resources at the spectrum level through implementation, and to perform analysis on different means of solving the isolation problem if these exist, and to examine the isolation problem in the context of different applications and scenarios.

We start by outlining the design principles which we desire from any implementation, and examine existing implementations to see how they satisfy these design principles. Concluding that no satisfactory implementation exists, we decide to develop an implementation ourselves.

We describe the design of our implementation, and perform proof-of-concept simulations in this chapter. However, as mentioned, the isolation problem can only be examined through real world testing, and therefore the following chapter is concerned with over-the-air experiments to determine how the isolation problem can be solved in practice.

6.2 Design Principles

As is the case with any implementation, it is useful to have a set of design principles that help us keep to our overall aims. Therefore, before examining existing implementations or developing our own we draw up a set of design guidelines, which we can follow to ensure that any implementation we use is truly virtualization and can be used to analyse the isolation problem.

It would be desirable to use an implementation that can be extended for use in other virtualization research, such as examining the embedding problem, or comparing virtualization methods for different scenarios and at different representation levels. The definition of virtualization which we developed previously can serve as a good guide to formulate design principles:

Virtualization is a resource mapping that occurs in the abstract domain. Virtualization takes places within any one representation level in the abstract domain. The representation level used is a design choice. Abstract resources before virtualization, known as real resources, are limited by the granularity of the underlying physical resources. Abstract resources after virtualization, known as virtual resources, are not, and can be larger or smaller than real resources. Virtual resources are independent and can be allocated simultaneously to multiple users, each with the illusion of full ownership of their resources. Virtualization can always be done recursively, since users perceive no difference between the virtual resources and the real resources.

We develop a number of key design concepts, several of which stem directly from the definition:

Design Principles
1. We would like to use existing implementations as much as possible to avoid reinventing the wheel. Similarly, we would like to be able to extend any implementation we use for further research later. As a result, open-source implementation appears to be more suitable than proprietary solutions. 2. Ideally, the implementation should cater for both splitting and combining real resources - meaning that it should be many-to-many. 3. Any implementation should be recursive in theory, even if overheads and other practical issues make recursion impossible. 4. Ideally, the isolation problem and the embedding problem should be separate; we should be able to use any isolation method and any appropriate embedding technique together. 5. As overarching theme, we would like the implementation to be as flexible as possible.

- 6 We would like the implementation to be as simple as possible, while retaining flexibility and high performance.
- 7 Ideally, we would like the implementation to be as cheap as possible, while also having high performance.

Although these design principles are somewhat obvious, they act as a useful guide for evaluating existing implementations or for designing our own implementation, so that we can see exactly how an implementation fulfils our design criteria.

6.3 Evaluating Existing Implementations

A number of implementations exist which might serve as a useful starting point for spectrum-level virtualization. Several works focus on spectrum agility, another term for Dynamic Spectrum Access (DSA), such as [161–164]. A common theme of these works is that they achieve spectrum agility using spectrum shaping and baseband processing operations.

There are also several implementations that concentrate on sharing an Radio Frequency (RF) front-end between multiple Physical Layer (PHY) signals, either by modifying the RF front-end hardware ([165]), or through software defined baseband processing [161, 166]. In contrast, the hardware-based implementation FlexRadio [167] can unify multiple RF front-end chains into a single wireless node.

However, the most comprehensive implementation of spectrum-level virtualization is the Spectrum Virtualization Layer (SVL) developed by Microsoft Research Asia [124]. The objective of SVL is to support dynamic spectrum access in general wireless networks, and to overcome problems that can arise from the allocation of fixed (channelized) spectrum bands, and the contiguous spectrum bands which most traditional wireless PHY techniques require. SVL enables both the sharing of an RF front-end between multiple PHYs and spectrum agility.

Therefore, it is worthwhile to examine SVL in detail, to evaluate whether we can use SVL to do analysis on virtualization techniques. To be consistent, we have used the language of virtualization theory in this examination, although this language is not used by the designers of SVL.

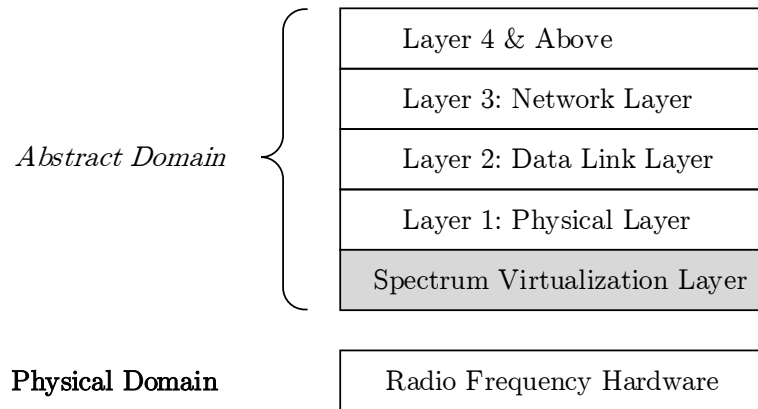


Figure 6.2: The Spectrum Virtualization Layer (SVL) sits under the physical layer of the OSI model.

6.3.1 Overview of SVL Architecture

Since SVL is designed to be a general virtualization technique, it is proposed as sitting below the physical layer in the Open Systems Interconnection (OSI) model of communication, shown in Figure 6.2. SVL presents the PHY with a *virtual* baseband, which is contiguous and of fixed width, a requirement for almost all existing PHY techniques. The virtual baseband is mapped to real spectrum by SVL, which can be of different width to the virtual baseband, or even consist of several non-contiguous bands. This mapping is achieved through real-time reshaping of baseband signals, in a way that is agnostic and transparent to the physical layer, allowing conventional PHY techniques to be used unchanged. In this manner, SVL uses *frequency* isolation to ensure the independence of each virtual link.

SVL provides each PHY with a virtual baseband, which is mapped to the real bands of the RF front-ends, shown in Figure 6.3. The width of a virtual baseband can be specified by each PHY during the initialization stage. The mapping of virtual bands to real bands is done based on a *spectrum map*, and the mapping is quite flexible, as virtual bands can be mapped to real bands with the same width or larger, or several non-contiguous bands. Spectrum allocation is governed by a *spectrum manager*.

The purpose of the spectrum manager is to monitor the current spectrum (based either on sensing, or by querying a database), and to allocate real spectrum bands to virtual bands. In other words, the spectrum manager deals with the embedding problem. As the focus of SVL is not on the embedding, the authors of the paper assume that SVL already has a spectrum map. The authors' focus is instead on enforcing the spectrum map, i.e. on the isolation problem. Thus by separating the isolation and embedding problems, SVL provides flexibility and modularity, and each problem can be tackled

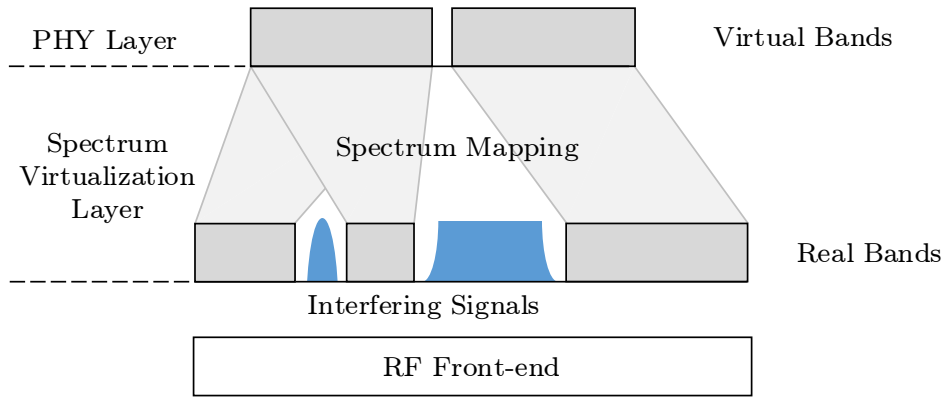


Figure 6.3: SVL maps virtual spectrum bands to real spectrum bands.

independently.

Thus the most important component of SVL is the *signal reshaper* which reshapes the virtual baseband signals to fit the real spectrum at the Transmitter (Tx) and performs the inverse reshaping operation at the Receiver (Rx). As the authors of [124] rightly point out, the signal reshaper should be both PHY agnostic and transparent to the PHY. The first condition implies that the signal reshaper only adopts generic Digital Signal Processing (DSP) operations without knowing any specific details about the signals. The second stipulation means that the PHY will not be able to notice any difference between the virtual baseband and the real baseband; in other words it is the property of recursion as seen previously.

The third condition that the authors introduce is the requirement of simplicity, that is for the reshaper to have as minimal overhead as possible. For this reason, SVL does not perform any wireless protocols such as equalization, clock timing recovery, etc. In addition to creating extra overheads, these protocols would be redundant, as the specific PHYs would perform such functions already, and moreover know the best means to achieve them. The specific signal reshaping operations will be discussed in detail in the next subsection.

As a final step, the reshaped signals are mixed together at the transmitter so that they can be sent to the RF front-end, while at the receiver the signals are split before reshaping. The designers of SVL rightly argue that the process of combining several PHYs to a single RF front-end is a form of RF front-end virtualization. However, SVL is only many-to-one in terms of RF front-end virtualization, and thus only many-to-one in terms of wireless link virtualization (see Section 4.3.2 in Chapter 4). The full architecture of SVL is shown in Figure 6.4.

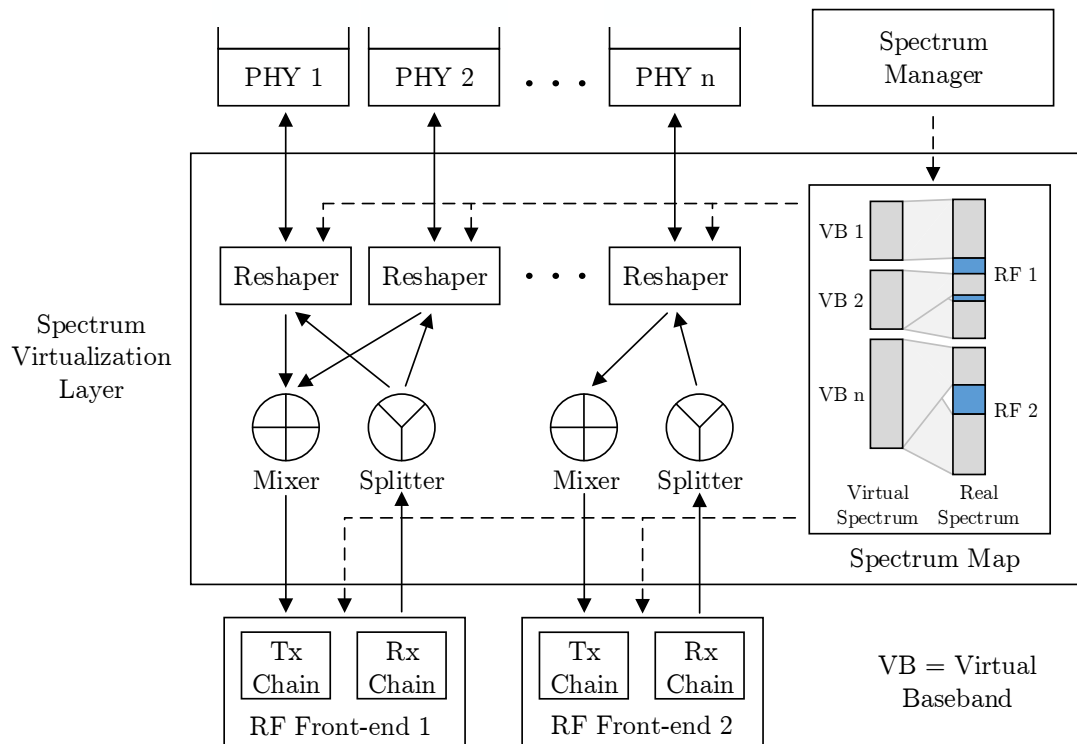


Figure 6.4: Full architecture of SVL. The mapping of virtual spectrum to real spectrum is performed by dynamically reshaping signals, based on the spectrum map. Solid lines indicate the data paths, while dashed lines show the control paths. The unusable parts of the real spectrum are shown in blue.

6.3.2 Signal Reshaping Operations

As mentioned, the signal reshaper is the core component of SVL. The reshaper consists of three operations: signal decomposition/recomposition, bandwidth and sampling rate adjustment, and frequency shifting. Figure 6.5 shows these three operations for the transmitter and receiver chains in SVL.

Signal decomposition at the transmitter can split a virtual baseband into several sub-streams, which can then be mapped separately to the real spectrum. Conversely, signal recomposition at the receiver combines these sub-streams into a single virtual baseband. The second operation, bandwidth and sampling rate adjustment, can change the PHY signal's bandwidth and sampling rate to match those of the RF front-end. Finally, frequency shifting moves the virtual baseband, centered at 0 Hz, to the associated real spectrum band at the transmitter, and vice-versa at the receiver.

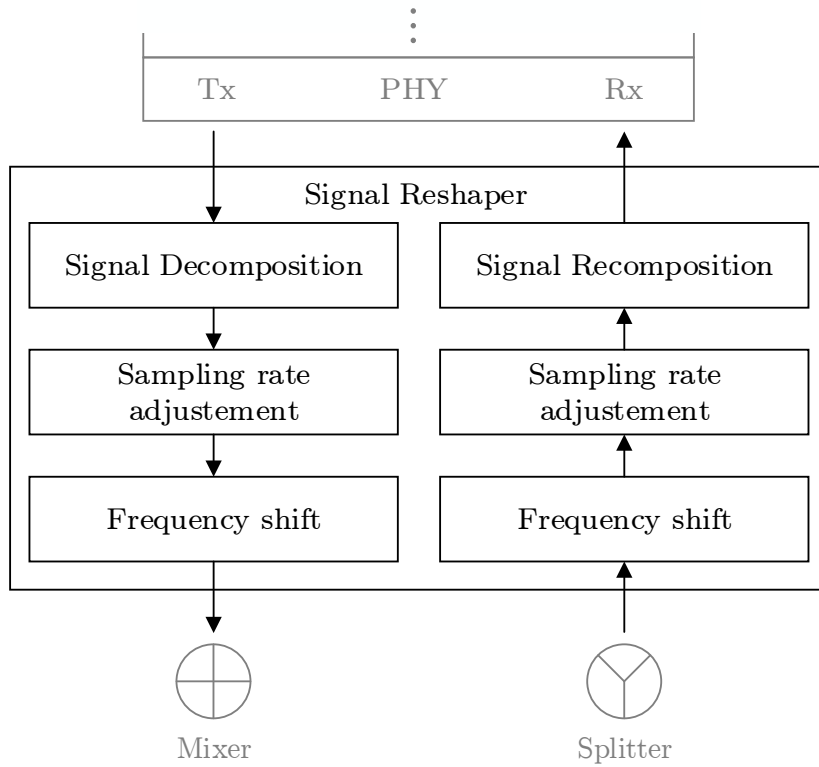
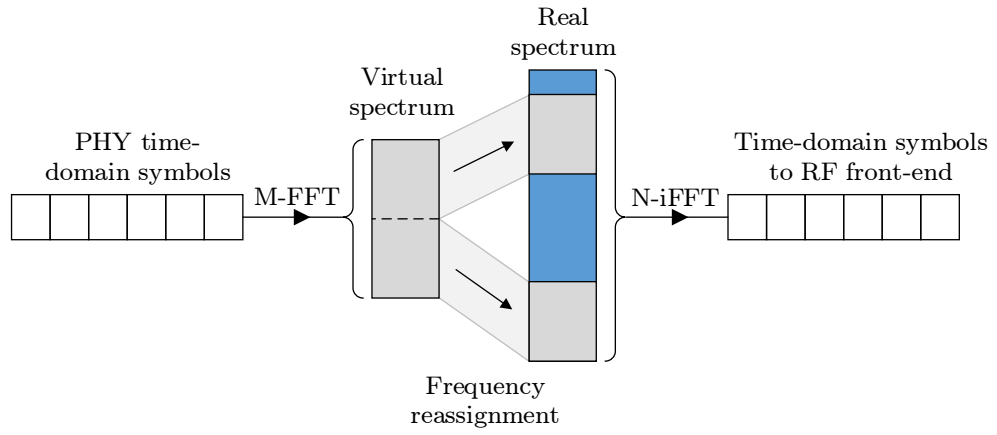


Figure 6.5: Components of the signal reshaper.

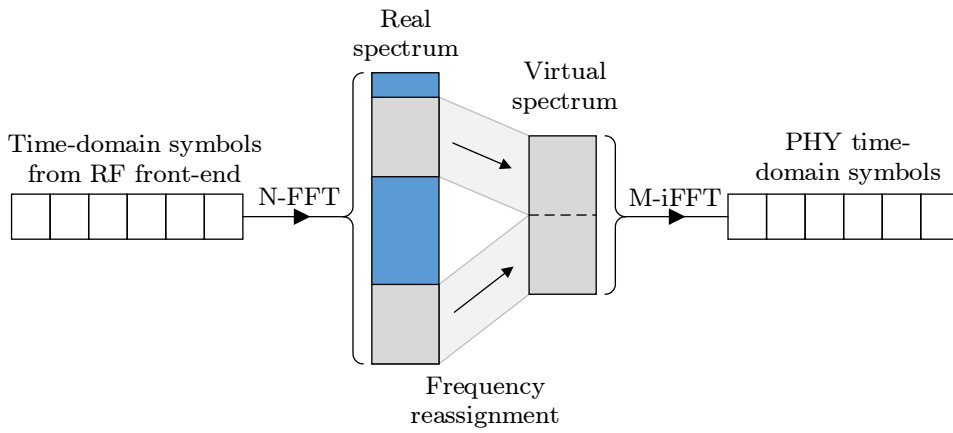
Signal Decomposition and Recomposition

Signal decomposition/recomposition is done on a symbol per symbol basis. Signal decomposition and recomposition cannot be done in the time-domain, since splitting a signal in the time-domain to multiple spectrum bands would mean that different symbols would undergo different channel conditions, violating the transparency principle of SVL. Instead, frequency-domain signal decomposition is used to ensure that each symbol experiences the same channel condition – although different frequency components of the signal pass through different sub-channels, they always pass through the same sub-channels, and thus the separate sub-channels appear to the PHY as a coherent wireless channel.

The steps of signal decomposition are shown in Figure 6.6(a). First an M -point Fast Fourier Transform (FFT) is performed on each symbol of the signal to convert the time-domain samples to the frequency-domain. Then the frequency-domain samples are reassigned to the real spectrum bands, and finally an N -point Inverse Fast Fourier Transform (iFFT) is done to convert the frequency-domain samples back to the time-domain. For signal recomposition, shown in Figure 6.6(b), the same steps are performed with the difference that an N -point FFT and M -point iFFT are done, and the real spectrum is mapped to virtual spectrum.



(a) Signal decomposition at the transmitter



(b) Signal recomposition at the receiver

Figure 6.6: Signal decomposition and recomposition in the frequency-domain.

There are several issues to consider for signal decomposition and recomposition.

1. In SVL, N is always greater or equal to M to ensure that the real bandwidth is never less than the virtual bandwidth. M is chosen to be a minimum of 64, and in the case of a multi-carrier modulated PHY, is set as equal to the number of sub-carriers to avoid introducing inter-carrier interference. N is set as the smallest power of 2 that is larger than or equal to M times the ratio of real to virtual bandwidth, i.e $N = 2^{\lceil \log_2(M \frac{\text{real bandwidth}}{\text{virtual bandwidth}}) \rceil}$.
2. SVL needs to know the number of samples per symbol, called K , to correctly map each symbol from virtual to real spectrum and vice-versa. By performing an M -point FFT and then an N -point iFFT, signal decomposition enlarges the bandwidth of the signal by a factor of $\beta = \frac{N}{M}$ and increases the number of samples to βK .

If the number of samples per symbol, K , is smaller than M , zero-padding is used to increase the number of samples to M , as zero-padding in the time-domain does

not change the frequency response of a signal. Then after the spectrum mapping and the N -point iFFT, N time-domain samples will be obtained, of which only βK are significant and are kept.

If $K > M$, then the steps taken to correct for this are slightly more complex. Firstly, an M -point FFT is done for every M samples, until L samples less than M remain. Next, an M -point FFT is done on the last M samples, i.e. the $(K - M)$ th sample to the K th sample. This last FFT window is shifted by $M - L$ samples, corresponding to a phase rotation in the frequency domain. The phase rotation is corrected for by rotating the phase on the corresponding frequency bins. Essentially this means that if frequency bin x is mapped to frequency bin y , then the sample at frequency bin y is multiplied by $e^{j2\pi \frac{M-L}{M}(y-x)}$. The last two groups of samples will overlap by $\beta(M - L)$ samples, and these are averaged to get the output.

At the receiver, similar operations are performed when K is different to M . However, signal recomposition will reduce the bandwidth by a factor of β , and reduce the number of time-domain samples from βK to K , thus cancelling out the bandwidth scaling caused by signal decomposition.

Sampling Rate Adjustment

This module adjusts the signal sampling rate to match the sampling rate of the RF front-end using re-sampling. Re-sampling is performed by interpolation and decimation. Let f_s be the sampling rate of the signal, and f_{RF} be the sampling rate of the RF front-end. First, the sampling rate of the signal is increased by interpolation to f_{LCM} , the lowest common multiple of f_s and f_{RF} . Interpolation is done by zero-padding $f_{LCM}/f_s - 1$ samples and passing the signal through a low-pass filter to remove imaging. Next, the signal is decimated by f_{LCM}/f_{RF} , to get the desired sampling rate of f_{RF} . At the receiver, similar operations are done to re-sample the RF front-end sampling rate to the signal sampling rate.

Frequency Shift

After signal decomposition the signal is centred at zero Hertz. The frequency shift module shifts the signal to the real spectrum bands by multiplying each sample by $e^{j2\pi f_{real}}$ where f_{real} is the central frequency of the real spectrum band and j is the imaginary unit number. At the receiver, the signal is shifted by $-f_{real}$ through multiplying each sample by $e^{-j2\pi f_{real}}$.

Additional Features

Some additional features are implemented in SVL such as the ability to map virtual bandwidths to narrower real bands. However, this is accomplished by increasing the time needed to transfer the signal, and artificially modifying the clock so that PHY layers are unaware of this. Since this could lead to timing issues in PHY and higher layer protocols, this aspect of SVL violates the transparency principle, and so we do not consider it.

6.3.3 Implementation

A prototype SVL has been implemented on Sora [168], a software radio platform that combines hardware acceleration and software techniques to achieve high-performance. Sora is highly optimized, for example using direct memory mapping, Single Instruction Multiple Data (SIMD) techniques and fixed-point processing. More details on Sora can be found in [168].

Synchronisation

When a signal passes through non-contiguous spectrum, that is when SVL maps a PHY to multiple real bands, it is necessary to align the time-domain samples at the receiver to correctly perform the FFT at signal recomposition. Therefore, SVL adds a synchronisation sequence to synchronise symbol boundaries.

The synchronisation sequence is generated by taking the first M samples of a pseudo-random noise sequence, performing frequency reassignment as if the M samples were frequency bins, and taking the N -point iFFT. The sequence is inserted as a header for a block of PHY symbols, i.e. for a whole PHY frame. The recomposition module knows the synchronisation sequence and continuously searches for this sequence, and when found synchronises the symbols.

SVL Experiments

The authors of [124] provide several experiments to evaluate the performance of SVL. First, they show that SVL can reshape the virtual bandwidth of different signals using different wireless PHYs (two types of IEEE 802.11b signals and an IEEE 802.11g signal) to match real spectrum allocation. In the same experiment, it is shown that non-contiguous spectrum allocation performs almost identically to contiguous spectrum

allocation. Another experiment shows that SVL can use spectrum dynamically and improve spectrum efficiency by using spectrum gaps in a non-contiguous manner. A third experiment shows that an RF front-end can be used independently by multiple PHYs with very limited impact, even when one PHY is transmitting and other is receiving.

There are several additional experiments that aim to measure performance issues. The first experiment shows that the error introduced by the FFT and iFFT operations increases with the number of FFT points, although increasing the fixed-point bit precision from 16 to 32 decreases the error substantially (from about -8 decibel (dB) to about -38 dB for an FFT size of 1024). However, the increase in precision comes at the cost of higher processing complexity. A second scenario measures the processing overhead of SVL.

6.3.4 Evaluation of SVL as Virtualization Research Platform

The Spectrum Virtualization Layer provides a very good implementation of virtualization at the spectrum level. Ideally, we could use SVL as a starting point for performing analysis on virtualization techniques (following the first design principle listed in 6.2), such as investigating which isolation methods are suitable for different applications, comparing virtualization techniques at the spectrum level with virtualization techniques at the information level, and developing metrics to measure flexibility, isolation, scalability, etc.

However, there are several reasons why using SVL on Sora might be difficult.

1. SVL is neither many-to-many in terms of RF front-end virtualization nor in terms of wireless link virtualization, and a many-to-many implementation would be significantly more useful as radios are increasingly being equipped with multiple RF chains. As the definition of virtualization reminds us, it is important not to limit virtualization to only sharing, as this ignores a large number of potential new applications and efficiency gains.
2. Until recently, and for most of the duration of this research work, Sora has not been open-source [169]. Even though Sora is now open-source, to the best of our knowledge, the Spectrum Virtualization Layer implementation is not available. This means that we would have to reimplement SVL in its entirety on Sora.
3. To use Sora, it is necessary to buy a radio control board, an RF front-end, and to have a multi-core computer with sufficient capabilities. These costs would add up

to about \$5k for a single node [170]. There are many software radios/RF front-end combinations which are much cheaper, and despite potentially having lower performance, these software radio and RF front-end combinations could be suitable for our purposes.

4. Because of the reasons listed above, there is no large user community with established methods of sharing implementations. The result is that there is no repository of pre-made components that we can reuse, and instead we would have to implement many components, which can take a lot of time.
5. There are several aspects of SVL which seem to be overly complex, such as the mapping of virtual bandwidth to narrower real bandwidth, necessitating modification of the hardware clock. Additionally, it might be possible to avoid many of the re-sampling operations if the bandwidths are chosen to align more closely with the FFT/iFFT bins.
6. The software radio platform that SVL is built on, Sora [168], is highly optimized and it would not be simple to extend SVL to use new isolation methods or to do other virtualization research.

In summary, although SVL implements spectrum virtualization using frequency isolation, it does not satisfy enough of our design principles listed previously in section 6.2. Therefore it was deemed necessary to develop our own spectrum-level implementation of virtualization.

6.4 Evaluating Platforms for Our Implementation

Although the Spectrum Virtualization Layer provides a complete solution, it does not suit our purposes of investigating the isolation problem. The next option is to create a new solution, but rather than building a completely new solution, aim to construct one from existing building blocks.

To this end, the most logical choice for our implementation appears to be to use some kind of software radio framework that would provide a library of existing blocks. Software radios have been developed with the purpose of providing flexibility and reusability by implementing radio waveforms and functionality in software, in contrast to radios in which processing is done by analog circuitry and digital chips [171].

While using an existing software radio would still require additional design, it would none-the-less, conform to the design principles set out at the beginning of the chapter.

There are several open source software radio frameworks which provide this type of option.

6.4.1 GNU Radio

GNU ⁶ Radio is a development platform that provides a repository of signal processing and radio protocol blocks which can be assembled to create software radios [172]. GNU Radio comes with a wide variety of instrumentation blocks and RF front-end interface blocks, allowing it to be used either as a simulation environment, or with front-ends for actual radio implementation.

One of the largest advantages of using GNU Radio is that a large community of hobbyist, commercial, and academic users exists, resulting in a large component base, and in significant educational material available [173]. There are many Out-of-Tree (OOT) modules [174] on the Comprehensive GNU Radio Archive Network [175], which can easily be installed using the GNU Radio framework. These modules add extra functionality to the base version of GNU Radio, for example as blocks that implement wireless standards such as Long Term Evolution (LTE) and WiFi.

GNU Radio Companion (GRC) is a graphical user interface for GNU Radio that enables easy experimentation and simulation [176]. It lists a library of components which can be added to a flowgraph and connected to create signal processing applications. A very good tutorial for GRC can be found at [177].

6.4.2 Iris

Iris is a software radio architecture for building highly reconfigurable radio networks, and can be used in conjunction with almost all RF front-ends [178]. Iris has predominately been used for cognitive radio research, such as sensing components, components for dynamically shaping waveforms, and frequency rendezvous applications [179].

6.4.3 LuaRadio

LuaRadio is a lightweight signal processing framework, aimed at rapid prototyping and embedded applications [180]. LuaRadio is written in the Lua language, which means that no compilation is required. However, LuaRadio appears to be a relatively new software

⁶The acronym ‘GNU’ is a recursive acronym meaning “GNU’s Not Unix!”

radio platform, which means that some functionality is lacking that exists in Iris and GNU Radio.

There are many other software radio platforms that exist, amongst which are OSSIE [181], SDR# [182], HDSDR [183]. However, these software radios are not as flexible or as commonly used as the software radios described above.

We choose GNU Radio for our implementation, mostly because it best satisfies our design principles, as it has a large number of existing components, is open-source, can be used with many types of RF front-end, and is relatively easy to use. These properties enable us to focus on implementing virtualization at the spectrum level, rather than spending time on ancillary features. Additionally, we can use the GNU Radio Application Programming Interface (API) to reuse existing components in our design to speed up development.

6.5 Designing Our Implementation

We design our implementation as an a out-of-tree module for GNU Radio, guided by the design principles outlined in section 6.2. The whole module is open-source and can be found online⁷. The implementation is designed to be many-to-many – both in terms of wireless link virtualization and in terms of RF front-end virtualization as detailed in Section 4.3.2.

Many-to-many RF front-end virtualization enables 1) the sharing of RF front-ends between multiple independent transmissions/receptions, 2) the aggregation of multiple RF front-ends for one transmission/reception, and 3) any combination of the sharing and aggregation (see Figure 3.2 in Chapter 3).

Similarly, many-to-many wireless link virtualization allows 1) multiple virtual links to share a real wireless link in some manner, 2) one virtual wireless link to exist over multiple real wireless links (although many-to-many RF front-end virtualization is required), and 3) some variation of of splitting and aggregation of wireless links.

Since our implementation uses many-to-many virtualization for both RF front-ends and spectrum, any type of wireless link virtualization is possible. Therefore, we call our implementation the Many-to-many Spectrum Virtualization Layer or MySVL.

⁷The code for MySVL can be found online at <https://github.com/jvandebelt/gr-mysvl>

6.5.1 Design Process

Following our design principles, we base the Many-to-many Spectrum Virtualization Layer (MySVL) on an existing implementation of spectrum-level virtualization, namely the Spectrum Virtualization Layer described previously. However, our implementation, MySVL, significantly differs to SVL in a number of respects. These key differences are outlined during the description of the design.

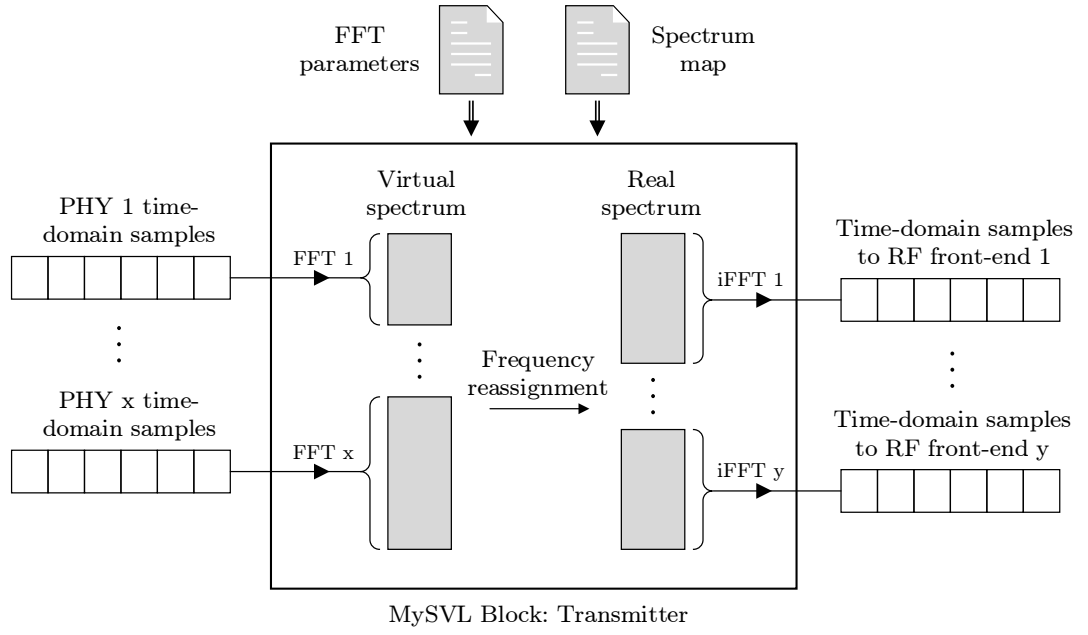
We developed MySVL over the course of about six months, and followed a number of iterative steps.

1. First, we conceptually designed the functionality of MySVL. SVL provided us with a good starting point, and it was relatively easy to see how we could improve on that design. The conceptual design of MySVL is described in detail in Section 6.5.2.
2. Next, we broke down virtualization tasks into a set of classes, and developed MySVL as a GNU Radio block. Again, this process was relatively straightforward as we could test each feature individually before bringing everything together. Implementation details are given in Section 6.5.3.
3. Having implemented the block, we then performed simulations in GNU Radio Companion to verify overall functionality, shown in Section 6.6. Based on these simulations, we changed several aspects of the block, but on the whole the implementation functioned as expected.
4. The final part of the implementation was to perform virtualization experiments over-the-air, explained in the next chapter, Chapter 7. The experiments showed that it was necessary to add additional components for synchronization, which we had hoped to avoid as we wanted to keep the implementation as simple and transparent as possible.

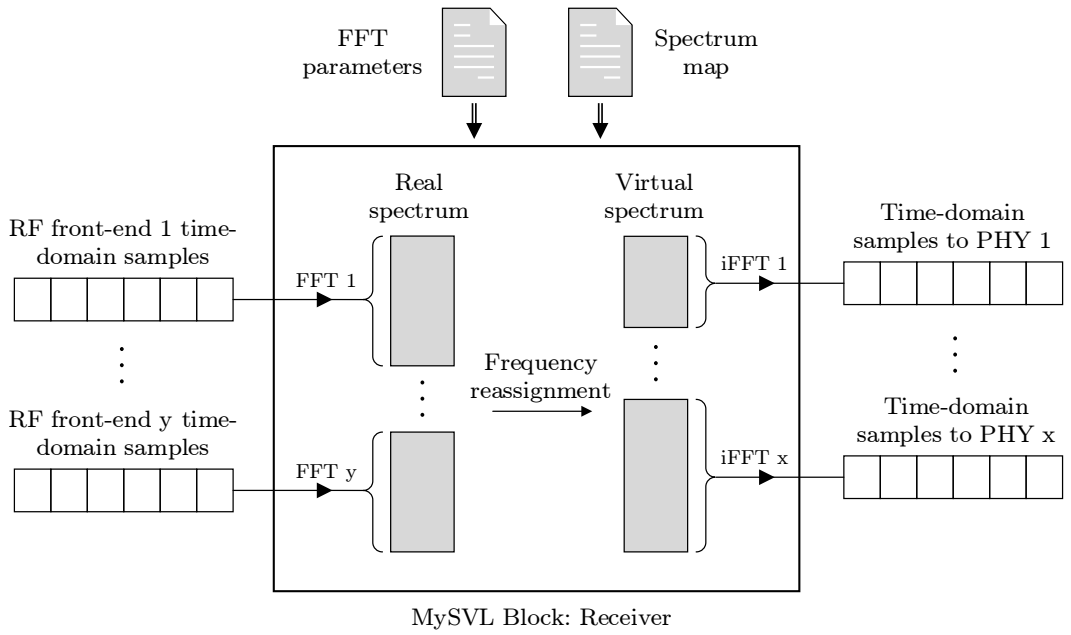
Although the design was conceptually not difficult to implement, the reality of the wireless medium provided the biggest challenge (see Chapter 7), as could be expected in any wireless link implementation.

6.5.2 Overview of the Resulting Design

The overall functionality of the MySVL block is shown in Figure 6.7. At the transmitter, the MySVL block takes in complex samples in the time domain from a number of physical



(a) Many-to-many spectrum virtualization at the transmitter. Multiple PHY layer signals 1 to x are mapped to RF front-ends 1 to y .



(b) Many-to-many spectrum virtualization at the receiver. The signals from RF front-ends 1 to x are mapped to PHY layers 1 to y .

Figure 6.7: Overview of MySVL functionality. Time-domain samples from multiple signals (regardless of whether these are from the PHY layer, or from the RF front-end) are converted to the frequency domain using FFTs – the size of the FFTs will depend on the signal. Next, the frequencies contained in the signals are reassigned based on a spectrum map, after which iFFTs convert the frequency domain samples back to time-domain signals. Note that the functionality of both the transmitter and the receiver is identical; only the FFT parameters and the spectrum map are different.

layer inputs, *which can be of any type of signal using any type of modulation*. The block performs an FFT on each input signal, and next remaps the input frequency bins (the virtual spectrum) to output frequency bins (the real spectrum). Virtual spectrum can be mapped to real spectrum either contiguously or non-contiguously. Then the real spectrum frequency samples are converted to the time-domain using and inverse FFT and forwarded to the radio frequency front-end.

The MySVL block at the receiver operates very similarly, except instead of taking in complex samples from the PHY, it receives samples from the RF front-end. Subsequently, samples are mapped from real spectrum to virtual spectrum, and outputted to the PHY.

As can be seen in Figure 6.7, the operation of the transmitter and receiver block is identical, only the spectrum mapping and the FFTs/iFFTs are different. Therefore noting this, we design our block in such a manner that both the transmitter and the receiver can use the same block, with different spectrum reassignment and FFT parameters. This simplifies our design significantly and additionally makes the MySVL block easier to use.

We design our block to read in the FFT parameters and the spectrum map from different text files; this increases flexibility and independence as it allows the spectrum map to be updated separately from the FFT parameters. In a sense, this is the separation of the isolation and the embedding problem, and the spectrum map can be generated using any embedding technique.

Following the design principles, we also omit much of the additional complexity that SVL implements, such as resampling signals, filtering signals, and splitting and mixing signals. All of this functionality already exists in GNU Radio, and can easily be used with our block if desired. In addition, there is no need to shift the baseband signals to the frequency of the RF front end as the front-end blocks and hardware drivers already perform this function. Thus we focus only on the essential spectrum virtualization operations to keep our implementation as simple and flexible as possible.

6.5.3 Detailed Design

We now describe the design in detail and how it is implemented in GNU Radio (also referred to as ‘gr’ from here on). We implemented MySVL as a series of C++ classes, using the GNU Radio out-of-tree framework. The class diagram for MySVL is shown in Figure 6.8, following the Unified Modelling Language (UML) standard [184].

In this diagram each box represents a class, with the name of the class shown in the top section. The middle section of the box lists the members of the class, while the

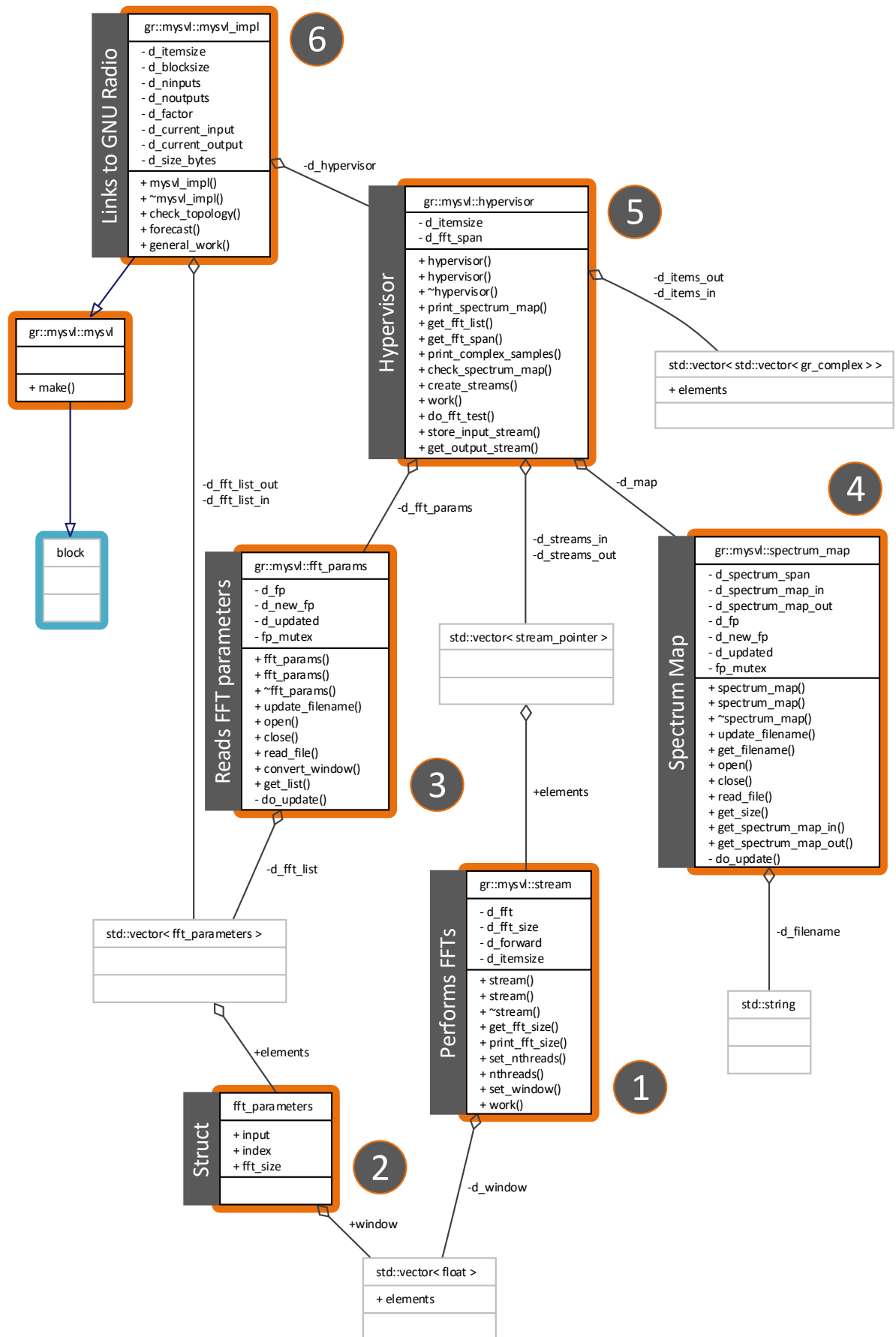


Figure 6.8: Class diagram for MySVL . We create the classes outlined in orange, while the other classes are part of the standard C++ library. The block class highlighted in blue is the GNU Radio superclass from which all blocks inherit.

bottom section of the box shows the methods of the class. A “-” indicates that a member or method is private, while a “+” signifies a public member or method.

The diagram also explains the relationships between classes. The blue arrows with pointed arrowheads signify the inheritance relationship, with the arrow pointing to the superclass. The arrows with hollow diamonds are aggregation links, pointing to the container class which has one or more members of the contained class’s type. These members are shown beside the aggregation arrows. For example, in the diagram we can see that the `gr::mysvl::stream` class (marked one) has a `std::vector<float>` member called `d_window`. More information about the types of relationships in class diagrams can be found in [184].

The rest of this section describes each class one by one, and how the classes fit together to create our overall implementation. We use the term stream from GNU Radio to describe a data flow in software radio. Streams pass from block to block and can be modified in some way by those blocks. In the case of MySVL, we can have a variable number of input streams of complex samples, and similarly a variable number of output complex sample streams.

1. Stream class

The purpose of the stream class is to perform the fast Fourier transforms for every stream of complex samples, whether these are forward FFTs in the case of input streams, or backward (i.e. inverse) FFTs for output streams.

The stream class uses the Fastest Fourier Transform in the West (FFTW) package but, following our design philosophy, we build the stream class on the `gr::fft` class. This requires linking to the `gr::fft` class when compiling MySVL, but fortunately there is some documentation on how to do this.

FFTs of any size can be done, set through the `d_fft_size` variable, and either in the forward or backward direction, controlled through the boolean `d_forward`. Additionally, the `d_window` variable allows the stream class to use different windows for the FFT such as rectangular, Hamming, Kaiser, or Blackman windows.

Using the stream class is very easy. After declaration, the `work` method can be called passing in input complex samples by reference in either the time- or frequency-domain (depending on whether the FFT is forward or backward). Output samples are also passed in by reference and the `work` method modifies these values to get the desired frequency- or time-domain stream.

1	1	1	8	rectangular
2	1	2	8	rectangular
3	1	3	8	rectangular
4	0	1	12	rectangular
5	0	2	12	rectangular

Figure 6.9: Example of an FFT parameters text file. In this case there are three input streams, each with an FFT size of 8, while there are two output streams of FFT size 12.

2. FFT parameters struct

In MySVL, each input and output stream can use unique FFT parameters such as size and window. Therefore, we declare a struct called `fft_parameters` to hold these values for each stream. The struct consists of four values; a boolean to show whether the stream is input or output; an index integer; an integer to hold the size of the FFT; and finally a vector of floating numbers to hold the window.

3. FFT parameters class

As mentioned, MySVL reads in the FFT parameters from a file. The FFT parameters class is used for this functionality. When the class is constructed, a filename argument can be passed in referring to a text file containing the FFT parameters. Alternatively, the filename can be given or updated using the method `update_filename`.

The filename is used to open the text file, and to read the values contained therein. Several error checks are performed, and a mutex [185] is used to prevent other programs from causing problems by modifying the file simultaneously. We use a vector of `fft_parameters` (the struct described previously) to store the values that are read in.

The format of the text file containing the FFT parameters is shown in Figure 6.9. Each row of the text file refers to an input or output stream.

A one in the first column indicates that the stream is an input, and a zero indicates that it is an output. The second value of each row is the index of the stream, and we track the input and output indices separately. Next, the size of the FFT for each stream is conveyed by the third column, and finally the last value gives the type of window for the FFT.

The FFT parameters class reads the first three values into the struct directly as a boolean, integer, and integer respectively. However, guided by the principle of reusing existing components as much as possible, we use the `gr::fft::window` class to instantiate

the vector of floats make up the FFT window. The window is of length `d_fft_size`, and the type is matched to the value from the file.

The interface for the FFT parameters class is easy to use. When the class is declared with a filename, the text file is opened and read immediately. Provided there are no errors, a vector of `fft_paramters` is generated. The `get_list` method can be used to access this vector. Finally, the `update_filename` method opens the new file and updates the vector of `fft_parameters` accordingly.

4. Spectrum map class

The overall functionality of the spectrum map class is to read in the spectrum map from a text file. The idea behind this is that the spectrum map would be generated by an independent embedding process (see Chapter 3), solving the embedding problem in some context, such as the embedding algorithms described in Chapter 8). The spectrum map class reads in the file in a similar manner to the FFT parameters class.

In addition, the text file can be reread without updating the filename, which means that dynamic embedding can be possible, meaning that the embedding can be changed for existing virtual resources (again, see Chapter 8 for a more full exploration of the embedding problem). Currently we have not developed the other features needed to implement this fully, but by allowing for this in the future, the design remains flexible.

Figure 6.10(a) shows the format of an example spectrum map text file. We choose to use frequency bins as our format for the spectrum map, instead of a bandwidth and central frequency as used in [124] (see Section 6.3). The advantage of using frequency bins directly is that there is no need to perform complex resampling and bandwidth adjustment techniques; however the trade off is that the virtual spectrum can only be of the same width as the real spectrum. However, if a signal is to use a larger or smaller bandwidth, it is possible to use existing GNU Radio blocks to resample a signal before passing it into MySVL at the transmitter, and to do the inverse resampling at the receiver. In this way, we keep the design of MySVL simple.

Each row of the text file represents a frequency bin. The first column represents the input spectrum (virtual at the transmitter and real at the receiver) and the second column represents the output spectrum (real at the Tx, virtual at the Rx). The value of each number tells us which stream is used for the frequency bin, and how the input spectrum is mapped to the output spectrum. For example in Figure 6.10, input stream one is used for the first frequency bin and it is mapped to output stream one.

To demonstrate more clearly how spectrum mapping is done, Figure 6.10(b) shows

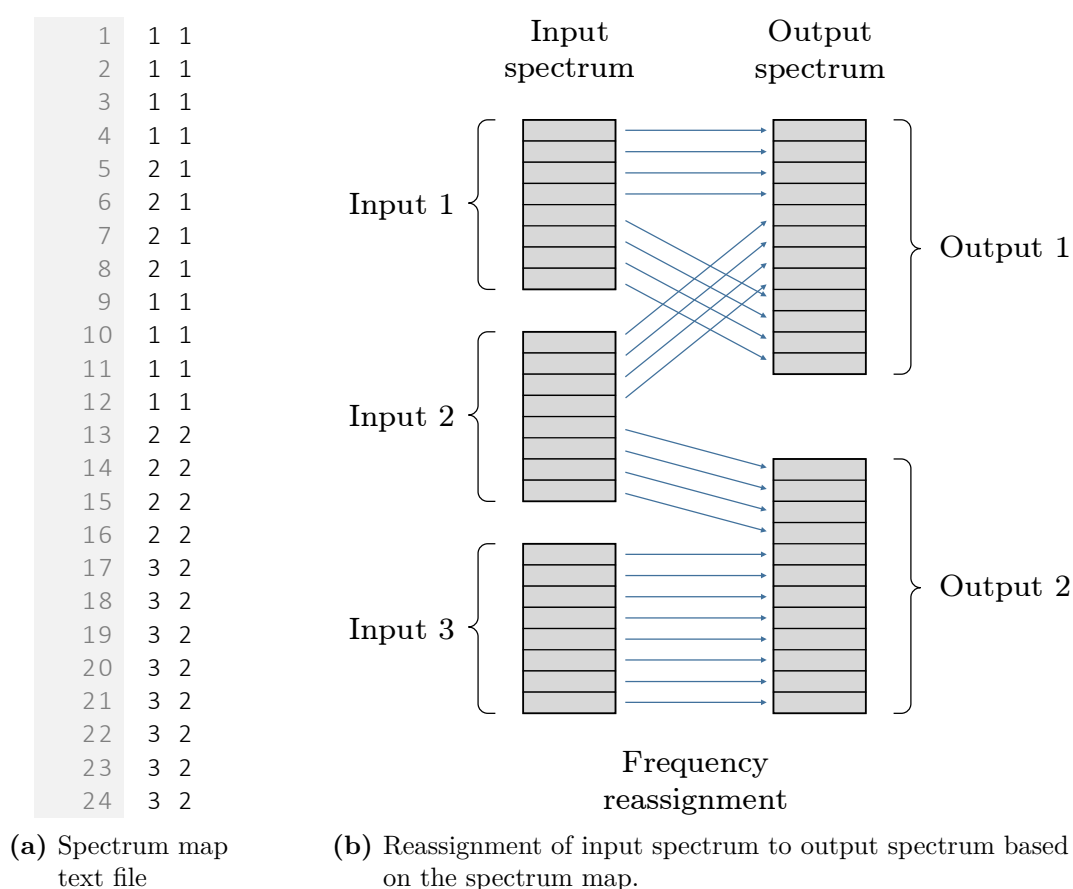


Figure 6.10: Frequency reassignment based on the FFT text file from Fig. 6.9 and the spectrum map text file.

the full frequency reassignment procedure using the FFT parameters shown previously and the spectrum map.

One aspect to be aware of is that the frequency reassignment is done sequentially for each input and output stream. In other words, the first frequency bin of input one is mapped when the first “1” appears in the input spectrum map. The second frequency bin of input one is mapped when the second “1” occurs. We design the mapping to happen in this manner, but it would be easy to change it.

The spectrum map class is used in a very similar way to the `fft_parameters` class. However, instead of passing a struct containing the FFT parameters, the spectrum map class has two methods called `get_spectrum_map_in` and `get_spectrum_map_out` which return a vector of integers containing the input and output spectrum maps respectively. Additionally, the text file can be reread dynamically.

5. Hypervisor class

In most forms of virtualization, such as process, machine, and network virtualization (see chapter 4), the entity which manages the virtual resources is called a hypervisor. The hypervisor is responsible for the isolation of virtual resources and the embedding of virtual resources to real resources.

In this context, our hypervisor class manages the virtual spectrum and brings together the components described thus far. It creates the virtual spectrum for each stream, ensures that no interaction can occur between streams, and maps the virtual spectrum to the real spectrum.

The hypervisor class contains several members that are classes which we described already, shown by the aggregation relation in Figure 6.8. Firstly, a `spectrum_map` called `d_map` is used to read in the spectrum map from file. Secondly, the `d_fft_params` member reads in the FFT parameters, and lastly a vector of type `stream` class is created for both the input and output streams. These vectors, called `d_stream_in` and `d_stream_out`, consist of a stream element for every input and output.

Other important members of the hypervisor are `d_items_in` and `d_items_out` which hold the time-domain input and output complex sample streams. These are the samples that continuously pass through the block as the signals that are transmitted and received. We describe how these samples are passed in and out of the hypervisor in the next subsection.

We now describe how the methods of the hypervisor and how they are used.

When the hypervisor is instantiated, the names of the FFT and spectrum map files are passed in as parameters to the constructor and the hypervisor uses the file names to instantiate the `d_fft_params` and `d_map` members. The constructor also obtains the list of `fft_parameters` and checks that the total FFT size of the inputs matches the total FFT size of the outputs.

The `create_streams` method populates the vectors `d_streams_in` and `d_streams_out` based on the `fft_parameters` list. It also ensures that the number of streams given in the FFT parameters text file matches the number of complex sample streams used in GNU Radio. The final function of `create_streams` is to check that the spectrum map and the FFT parameters are consistent, i.e. that the total number of rows for each stream matches the FFT size. It makes use of the `check_spectrum_map` method to do this.

During routine operation the hypervisor makes use of three methods to do spectrum virtualization. The first of these is the `store_input_stream` method which simply takes in a number of time-domain complex samples and passes them to the relevant input

stream. Secondly the work method is used after each input stream has received samples. Finally, the `get_output_stream` method returns the time-domain samples to the relevant output. Although each of these methods is simple to use individually, it is important use the methods correctly and update each stream accordingly.

In essence, the work function performs the desired functionality of the MySVL block, and brings together the separate components. It starts by converting each input stream to the frequency domain using the FFT and subsequently normalises the frequency-domain samples. Then the work method performs the spectrum mapping by sequentially copying each input frequency-domain sample to the corresponding output frequency domain sample, such as shown in Figure 6.10. Lastly, an iFFT is done for each output stream to change the samples to the time domain.

One of the most important points to keep in mind is that MySVL performs the FFT and iFFT on the samples of each stream, without any consideration of the samples per symbol. It is up to the user to set the FFT parameters appropriately, and although this might make MySVL slightly harder to use, it reduces complexity significantly and also provides flexibility. In a later section we show the impact of setting FFT parameters appropriately.

6. MySVL class

Although the hypervisor class performs the majority of the virtualization work, an additional class is needed to interface appropriately with GNU Radio. The MySVL class is a GNU Radio out-of-tree module block. The class was created with the aid of the `gr_modtool` which creates all of the files necessary to make a gr block. It also provides the C++ “boilerplate” code, which refers to code that is necessary in many places but is almost identical.

The overall purpose of the MySVL class is to create a flexible `gr::block` that uses our hypervisor class to do spectrum level virtualization. We now describe the class and how we use the methods and functions provided by GNU Radio namespace. The full documentation for `gr::block` is available at [186].

The MySVL class has a significant number of members but the most interesting ones are the hypervisor `d_hypervisor`, and the input and output vectors of `fft_parameters` `d_fft_list_in` and `d_fft_list_out`.

The MySVL constructor takes in the filenames of the FFT parameters and spectrum map text files, which are then used to create the hypervisor. Next, the constructor gets the vectors of `fft_parameters` from the hypervisor. The `fft_parameters` are needed for

many purposes in the MySVL class, most importantly for knowing how many samples to pass to the hypervisor from each input stream, and how many to return from the hypervisor to the output streams. GNU Radio provides a function called `set_output_multiple` which we use in the constructor to ensure that when we receive samples to process (in the `general_work` function described below), the samples arrive in multiples of the total FFT span. The MySVL block runs smoother and efficiently as a result.

The next method we use is the `check_topology` method which checks that the number of inputs and outputs of the `gr::block` matches the inputs and outputs given by the FFT file. This block is also used to run the `d_hypervisor.create_streams` method to setup the streams.

Another initialization method is the `forecast` method that estimates the number of input samples needed per stream to produce a desired number of output samples. In our case the number of samples needed depends on the size of the FFT of each stream.

The main method of MySVL is the `general_work` function. The parameters for this function are a requested number of output items, the number of input items given, and data buffers containing the input complex samples and locations to put the output complex samples. Since in our design the hypervisor performs most of the virtualization work, `general_work` passes the input samples to the hypervisor for each stream in turn, and performs the `d_hypervisor.work` function when complete. Subsequently, the output samples are returned from the hypervisor and put into the output data buffer. These steps are repeated until the requested number of output items have been fulfilled.

Although the operation of the MySVL class is relatively straightforward, one interesting point to note is that samples from each stream can be parsed at different rates, which to the best of our knowledge, very few blocks if any in GNU Radio do. The availability of the GNU Radio API has allowed us to program faster and more efficiently by reusing existing components.

GNU Radio Companion

We added the MySVL block to GRC by modifying the template eXtensible Markup Language (XML) file generated by `gr_modtool`. Essentially, the block passes in the required parameters to the MySVL class, and uses the GRC environment to make it easy to set the parameters and to connect MySVL to signal blocks. We use the MySVL block in GRC to do proof-of-concept simulations, described in the next section.

6.6 Proof-of-Concept Simulations

We now run several simulations to show that MySVL is functioning correctly, and to demonstrate spectrum level virtualization for several scenarios. We show that MySVL can be recursive as needed by the definition of virtualization and also measure the overhead of MySVL in terms of complexity.

These simulations are done using GNU Radio Companion, which allows us to quickly and easily create flowgraphs. For these simulations we are only interested in the functionality of the MySVL block, and therefore we assume ideal channel conditions.

6.6.1 Simulation 1: Understanding Spectrum Mapping

The first simulation we do is to test that spectrum mapping is happening correctly, under different spectrum map conditions. Figure 6.11 shows the flowgraph for Simulation 1. This flowgraph shows the various signal processing blocks in GRC, and how data passes between blocks. Data flows of bytes are shown in pink, while complex samples are in blue.

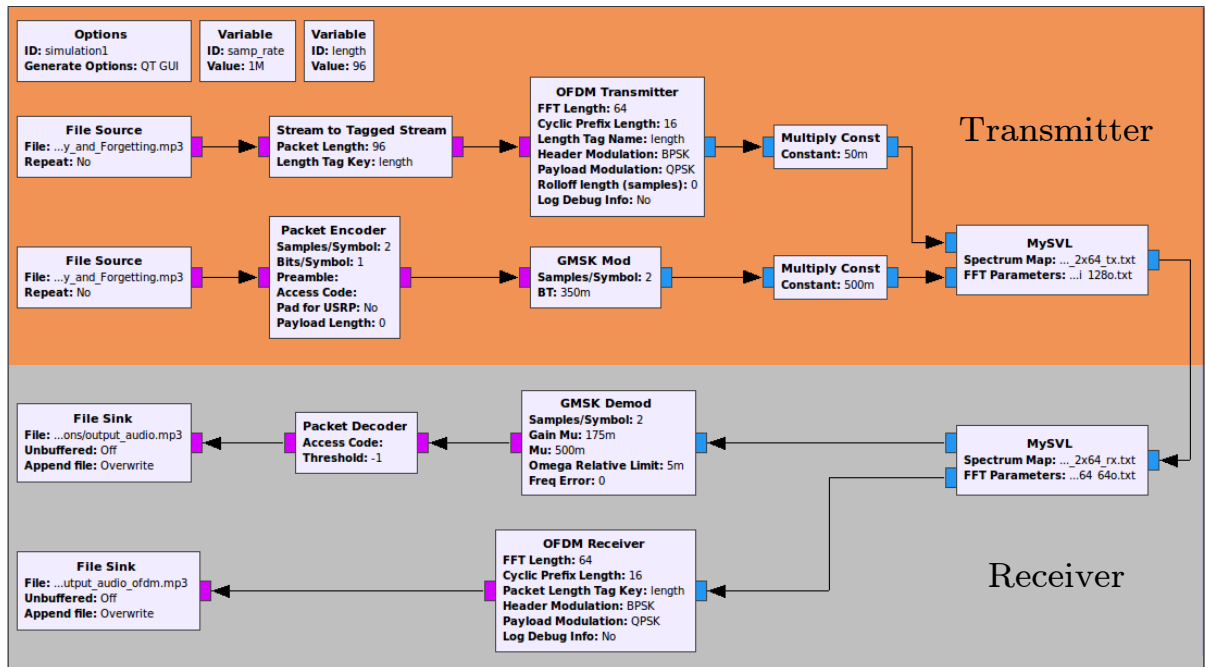


Figure 6.11: GNU Radio Companion flowgraph for Simulation 1. Two PHY signals, a GMSK signal and an OFDM signal, both use virtual spectra of 500 kHz. We choose these types of modulations but any modulation/ technology can be used in theory. The transmitter is shown in the top half of the graph, while the receiver occupies the lower half. MySVL maps the virtual spectra to real spectrum of 1 MHz at the transmitter, and performs the inverse virtualization at the receiver. We measure the frame error rate of the transferred files.

In this simulation, we use two different physical layer signals to transfer files. These signals could be of any type and modulation, but we choose a narrowband Gaussian Minimum Shift Keying (GMSK) signal and a wideband Orthogonal Frequency Division Multiplexing (OFDM) signal. These signals represent narrowband and wideband signals, but we expect results to hold for all types of signals as MySVL is designed as a generalised virtualization block.

The MySVL blocks are setup differently at the transmitter and the receiver. At the transmitter, the block takes in two inputs and has one output, while at the receiver the inverse occurs. The FFT parameters and spectrum map used also differ for the Tx and Rx.

Note that only simple PHY signals are used for the simulations, and we do not perform any higher layer protocols such as error correction and addressing. Therefore if any error occurs within a packet due to the MySVL block or otherwise, the entire packet is dropped. We measure the frame error rate based on the number of correctly received packets and compare this to the non-virtualized case.

We run the simulation with four different spectrum maps, 1a – 1d, but with the same FFT parameters. For all spectrum maps, both signals are assigned a virtual bandwidth of 500 kHz, and the real spectrum consists of 1 MHz. The difference in spectrum maps is in the frequency assignment; the first map assigns frequencies contiguously, while the other maps are non-contiguous.

For each simulation, we only show the spectrum mapping at the transmitter, since the mapping at the receiver is simply the inverse.

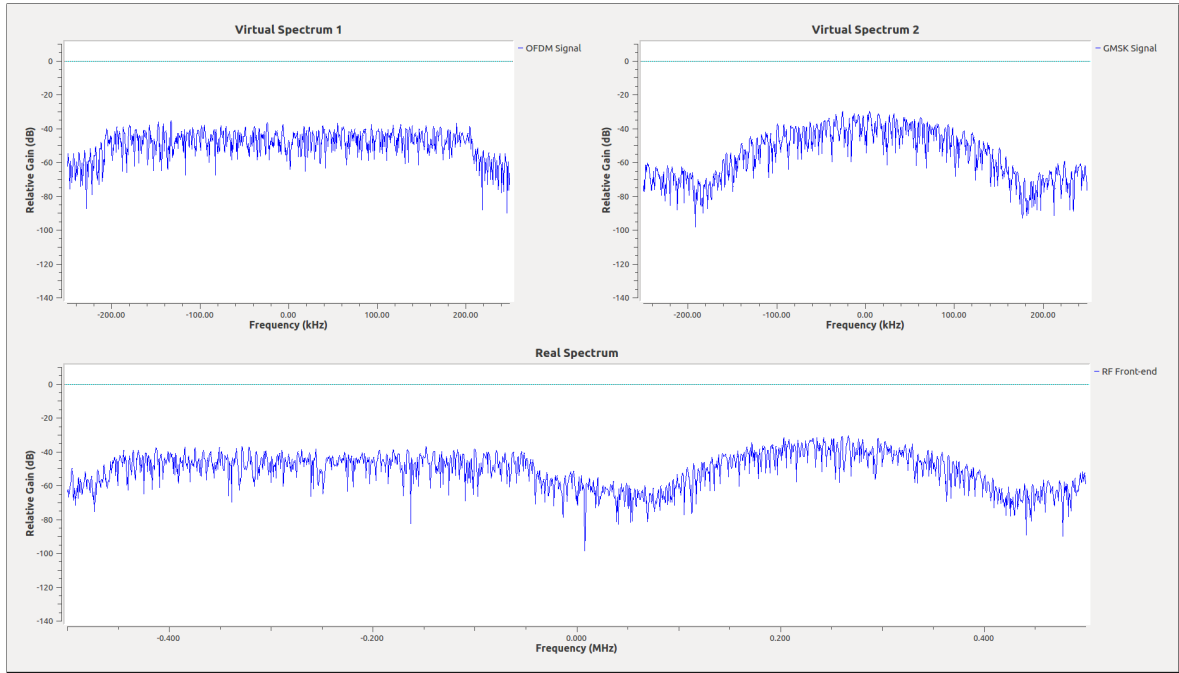
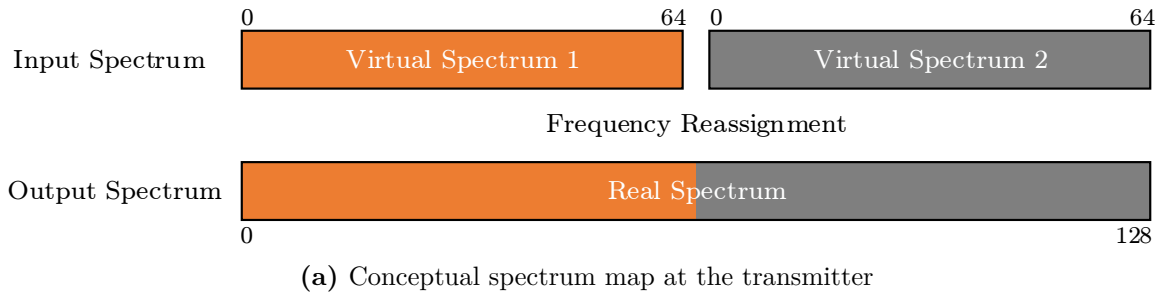


Figure 6.12: Simulation 1a: virtual spectrum is mapped to real spectrum contiguously, meaning that virtual spectrum is not split.

1a) Contiguous Spectrum Map

Spectrum map 1a shown in Figure 6.12(a) maps the OFDM signal to the lower half of the real spectrum, and the GSM signal to the upper half at the transmitter. The resulting virtual and real spectra are shown in Figure 6.12(b). From this figure, we can conclude that MySVL allows the two signals to share a single RF front-end, and that the mapping is occurring as desired.

We also measure the number of bytes received at the receiver and compare this to the baseline case without virtualization to check whether MySVL introduces any errors. For both cases the number of packets received is identical, and from further visual inspection of the files in detail we can confirm that MySVL does not introduce any errors.

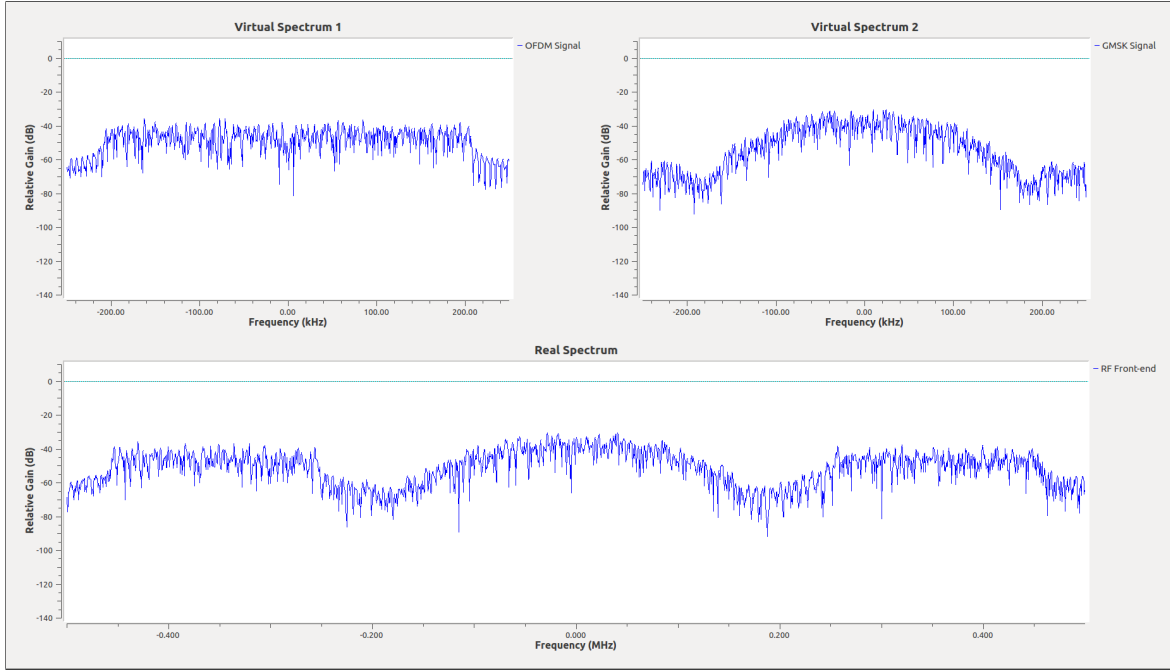
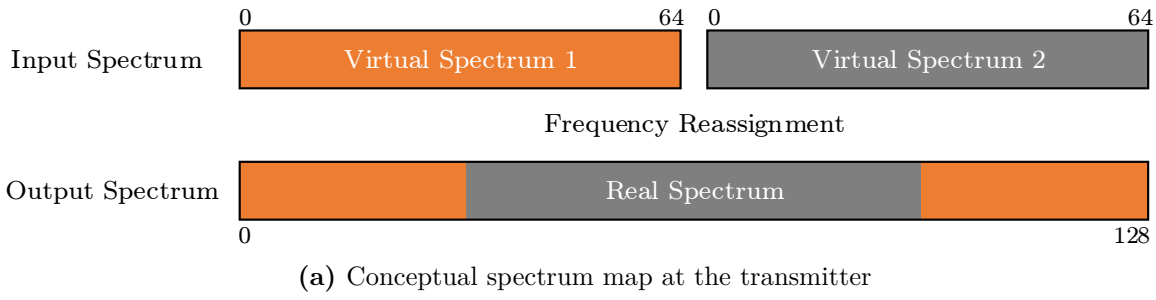


Figure 6.13: Simulation 1b: Spectrum mapping with split OFDM signal.

1b) Split OFDM Signal

The second spectrum map, 1b, tests the non-contiguous case where the OFDM signal is split, shown in Figure 6.13. As can be seen from this figure, MySVL can perform non-contiguous mapping correctly.

Again, we measure the bytes received for both data streams, and the total number of bytes received is the same as the base non-virtualized scenario and Simulation 1a.

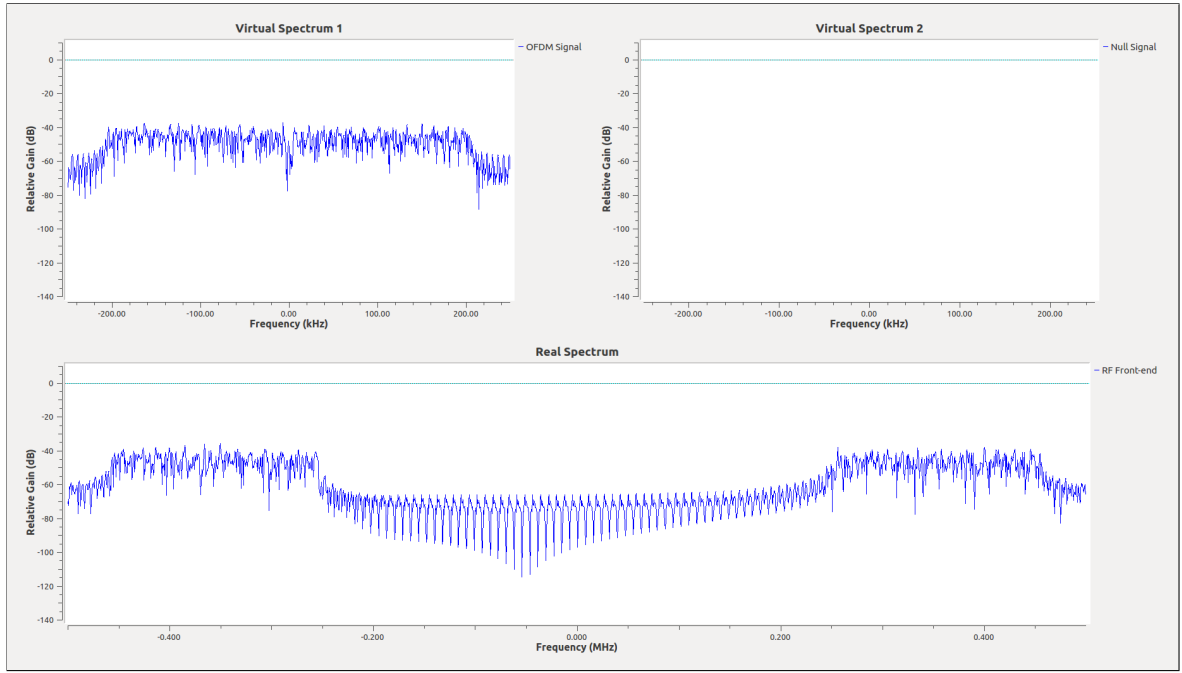
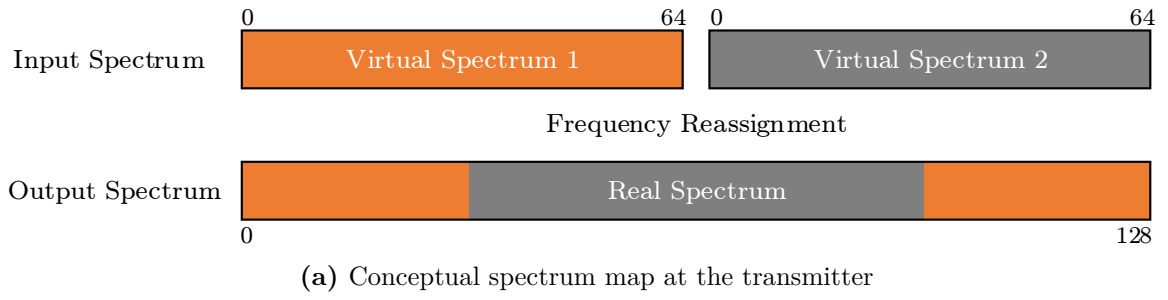


Figure 6.14: Simulation 1c: Spectrum mapping with split OFDM signal and null signal.

1c) Split OFDM Signal with Null Signal

The next scenario, 1c, introduces the concept of null signals, which can be used to avoid certain parts of the real spectrum. The flowgraph for this experiment is almost the same as previously, except that a null source and a null sink are used instead of the GMSK signal. The spectrum mapping is shown in Figure 6.14. This mapping could be used to avoid an interfering signal occupying the center of a channel.

Similarly to the previous simulations, there are no errors introduced by MySVL and the same number of packets are received as in the baseline scenario.

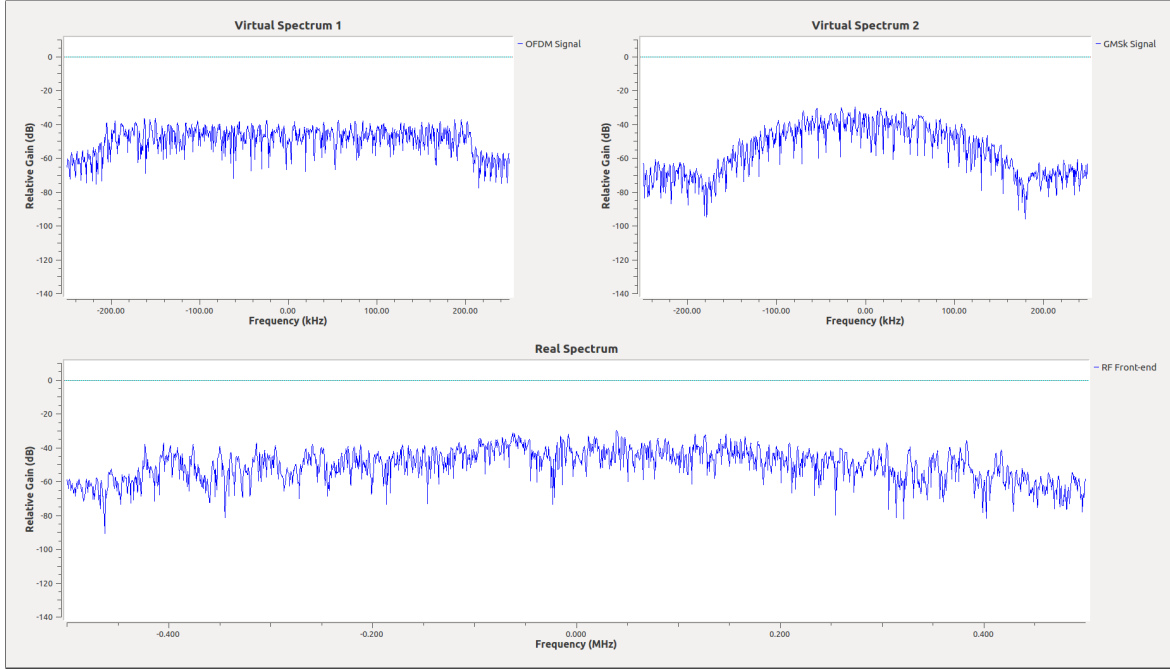
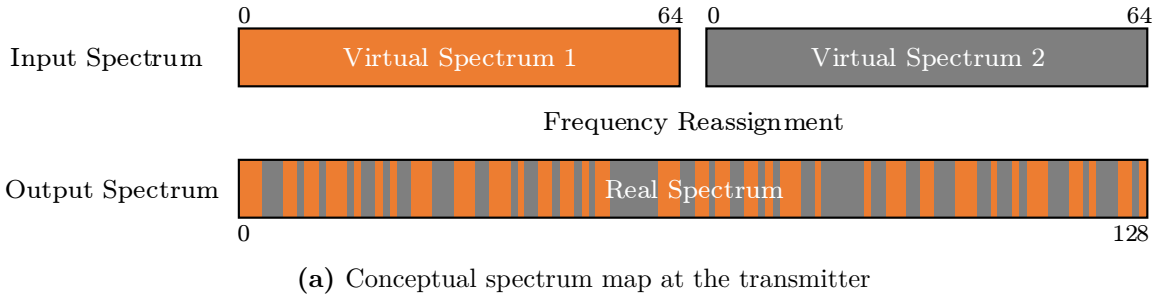


Figure 6.15: Simulation 1d: Spectrum mapping using a randomly generated spectrum map.

1d) Random Spectrum Map

The final spectrum map we test assigns virtual spectrum to real spectrum randomly, shown in Figure 6.15. This test is done purely as proof-of-concept, to show that any type of spectrum mapping is possible without any problems or errors introduced and that no packets are dropped, despite the complex spectrum map.

6.6.2 Simulation 2: Proving FFT Independence

The second simulation we run tests whether we can use flexible FFT sizes without introducing any overhead. We want to ensure MySVL is not introducing errors when used with signals that have a number of samples per symbol that is smaller or larger than the FFT size. Therefore we run a test with the same flowgraph as before (in Figure 6.11) but

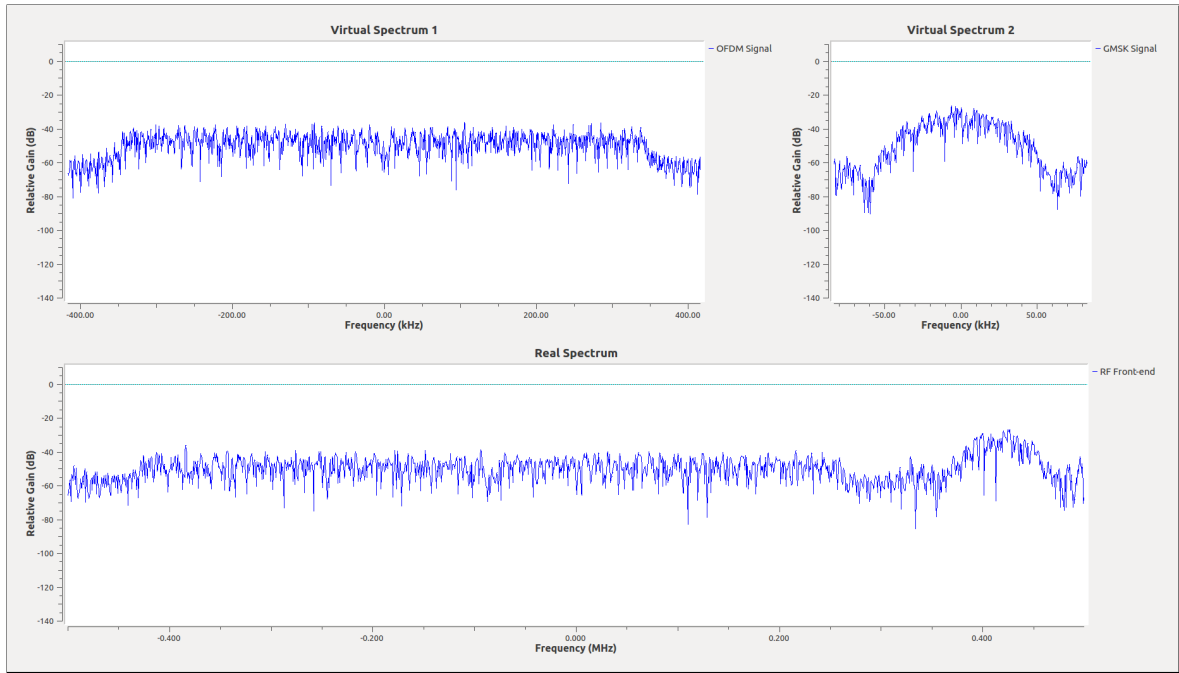
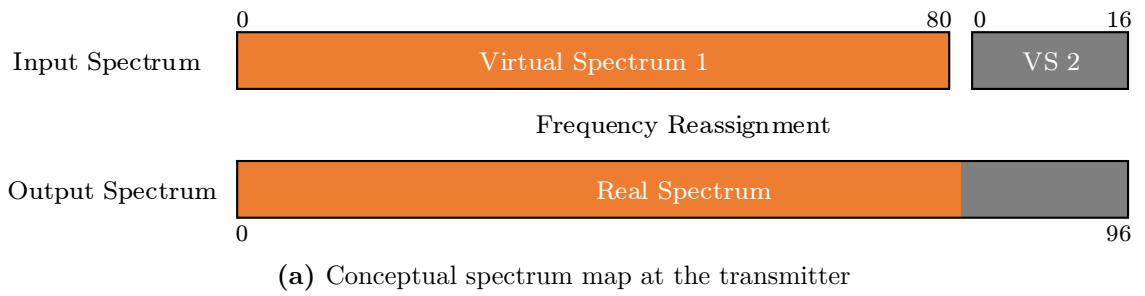


Figure 6.16: Simulation 2: spectrum mapping with flexible FFT sizes.

which uses an FFT size of 80 for the OFDM signal and an FFT size of 16 for the GMSK signal.

From Figure 6.16 we can see that flexible FFT parameters are possible. Again, examining the received files shows no errors introduced by the MySVL process.

6.6.3 Simulation 3: Testing Recursion

Simulation 3 checks that MySVL is recursive and that it satisfies our test for virtualization developed previously in Section 5.2. The flowgraph for the simulation is shown in Figure 6.17. We use the same notation as in Section 3.3 when discussing recursion.

In this scenario, the first instance of virtualization splits the real spectrum, measured as 512 frequency bins, into two virtual spectrum bands of 448 frequency bins and 64

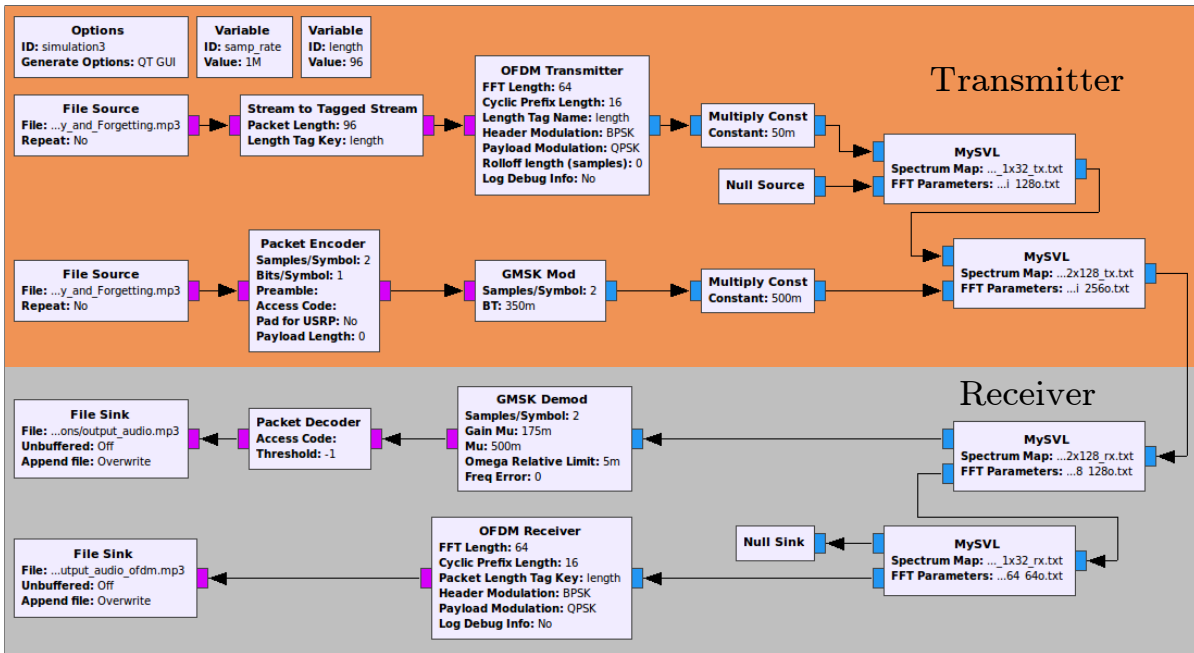


Figure 6.17: Flowgraph for Simulation 3. MySVL can be used recursively to perform multiple instances of virtualization. In the first instance of virtualization, an OFDM signal is split by a null signal. In the second instance, the resulting signal is combined with a GMSK signal. At the receiver, inverse recursive virtualization occurs.

frequency bins. The second instance of virtualization splits the 448 frequency bins into virtual spectrum of 256 and 192 frequency bins.

An OFDM signal is given the spectrum of 256 frequency bins, and it is split by virtualization instance two, similarly to Simulation 1c. The first instance of virtualization map this signal and a narrowband GMSK signal contiguously to an RF front-end. Figure 6.18 shows the conceptual spectrum map and the resulting virtual and real spectra.

We omit showing the second instance of virtualization as this is similar to that shown in Simulation 1c. From the resulting real spectrum, we can see that recursion is working as desired.

Again we note that the files received were as expected and that no errors were introduced by MySVL.

Practical Uses of Recursion in Wireless Link Virtualization

Although recursion is not thought of as a particularly useful property in the context of wireless link virtualization today, perhaps that could change in future wireless networks. There are increasingly more examples of recursion being used practically in virtualization in other domains, such as the use of recursion in node virtualization, or even the practice of Virtual Private Network (VPN) *cascading* or *chaining* [187]. In this case,

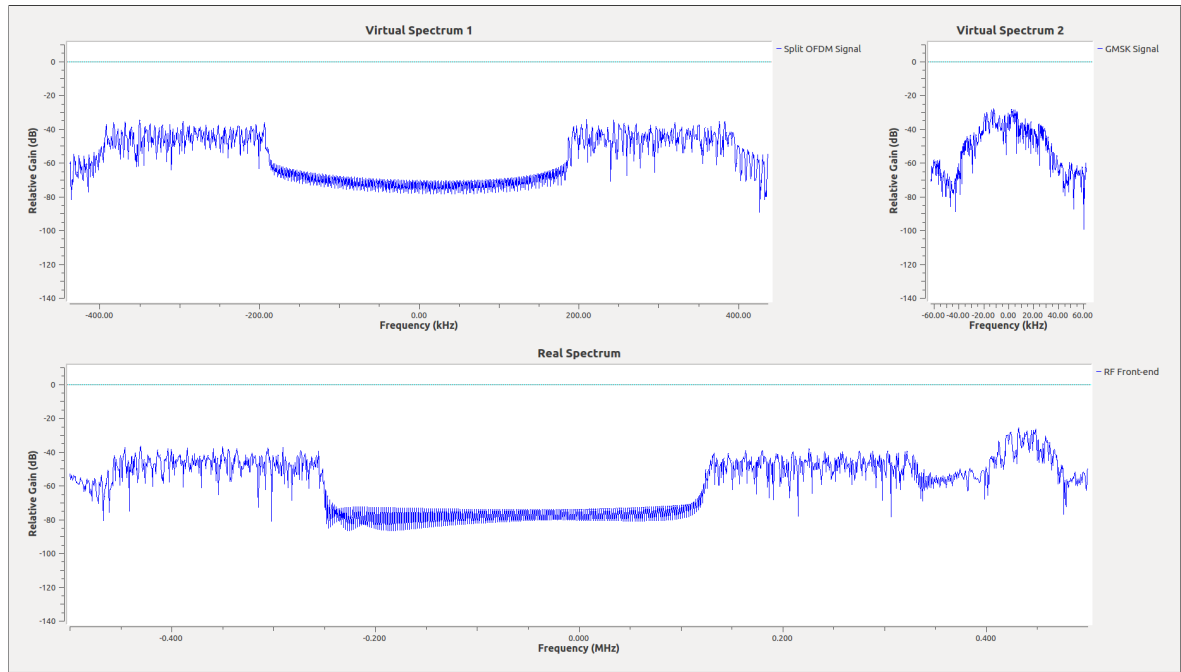
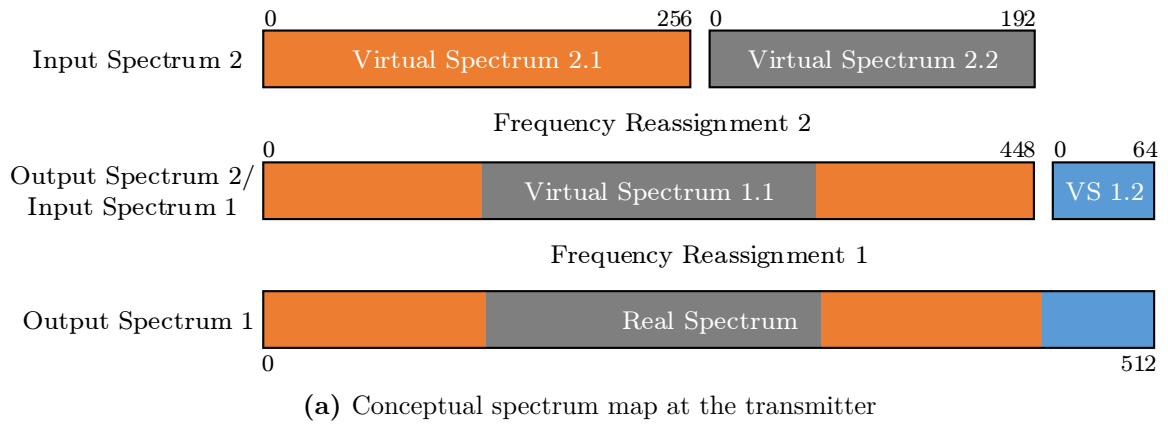


Figure 6.18: Simulation 3: Recursive spectrum mapping showing the second instance of virtualization. Virtual spectrum 1 is the output of a previous virtualization instance that maps an OFDM and a null signal to virtual spectrum 1.

multiple VPNs are used recursively to add additional layers of security compared to a traditional VPN. Recursion could similarly be applied to wireless link virtualization at the information level, providing extra security. Additionally, recursion allows users to set their own specific policies at the information level, such as customised packet inspection and filtering, rate control, and/or prioritisation.

Recursion could be used for spectrum-level link virtualization to enable a high degree of flexibility, since it allows the real spectrum to be split and combined in any number of ways. In addition, it allows the users of the virtual spectrum to quickly scale the virtual resource to their needs, as they can aggregate or split their virtual resources

again if desired. Recursion could be particularly useful for content delivery networks, as it could enable secondary markets in spectrum, allowing users to both buy and sell virtual resources. The adoption of such a system could lead to many new services and potentially reduce the cost of wireless telecommunications, as spectrum resources might be used much more efficiently, as many works on DSA have argued (see Subsection 7.9.4 and the surveys [188–190]).

6.6.4 Simulation 4: Verifying Many-to-many Operation

The final simulation we run is to test the many-to-many scenario, to verify that MySVL is operating correctly. In Simulation 4, we use the same many-to-many example seen previously in Section 6.5.3 in Figure 6.10. However, the FFT sizes used for the simulation are eight times larger.

As this test is simply to check whether the many-to-many case is working, we use an OFDM signal and two GMSK signals. The full flowgraph is found in Figure 6.19 and shows that the MySVL block can be used with a variable number of inputs and outputs.

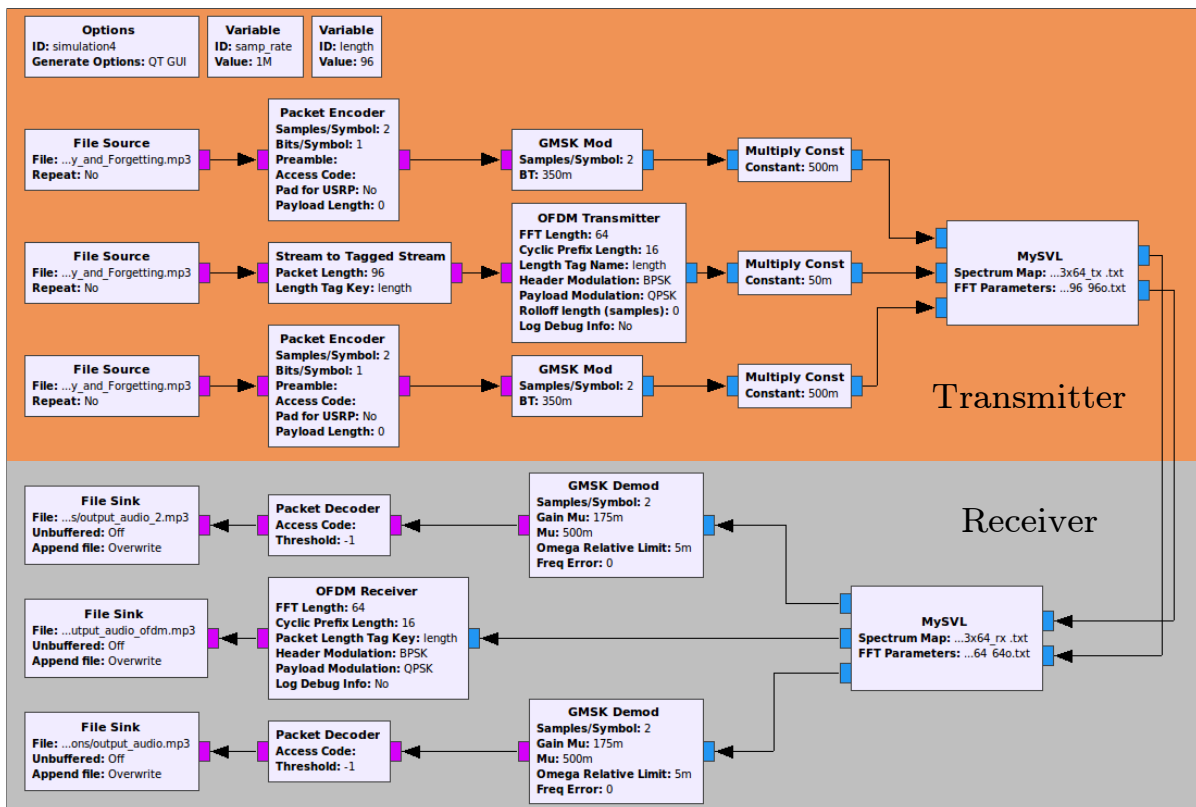


Figure 6.19: Flowgraph for Simulation 4. In this simulation, we map multiple signals to multiple RF front-ends, demonstrating that MySVL can do many-to-many virtualization. Two GMSK signals and one OFDM signal are mapped to two RF front-ends, which can then transmit the signals over the air independently.

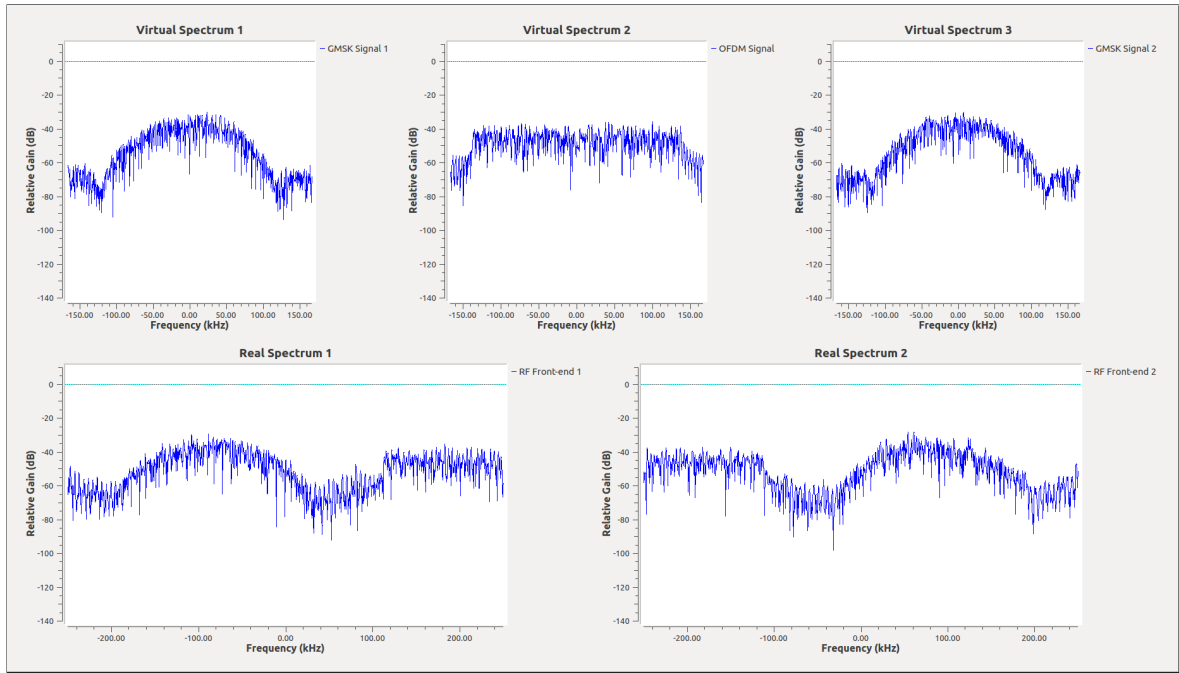
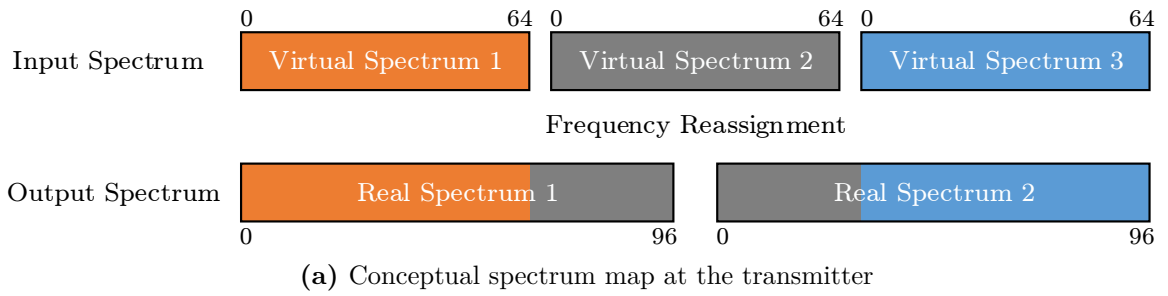


Figure 6.20: Simulation 4: Many-to-many virtualization allows signals to be split between multiple RF front-ends. This provides another method to avoid interference with other signals in addition to the use of null signals.

The resulting real and virtual spectra are shown in Figure 6.20, and as before, the files are received correctly. It is interesting to note that the many-to-many case presents another method of splitting a signal, such as the OFDM signal in this case, which can then be transmitted in separate parts of the real spectrum by the RF front-ends. Similarly to the use of null signals, this can be used to avoid already existing signals.

6.6.5 Conclusions

These simulations have demonstrated that MySVL works as desired in an ideal environment. In Simulation 1, we saw that any type of spectrum mapping can be used and Simulation 2 showed that the FFT size can be flexible without issue. Simulation 3 proved that MySVL passes the recursive test for virtualization, and showed some interesting func-

tionality. Similarly, Simulation 4 verified that MySVL is many-to-many, which further extends its possible applications.

6.7 Summary

In this chapter, we have developed an implementation of wireless link virtualization at the spectrum level of representation, to investigate the isolation problem. We designed the implementation with several aims in mind, the most important aims being flexibility and re-usability. To this end, we designed MySVL to be many-to-many in terms of both spectrum virtualization and RF front-end virtualization, meaning that multiple RF front-ends can be combined/split if desired.

Similarly, we built MySVL on top of the widely-used software radio framework GNU Radio, which is open-source and allows a large variety of differing RF front-ends can be used, ranging from cheap digital TV receivers to expensive high performance equipment. Many radio protocols and techniques are already implemented or exist as OOT modules, which allows for relatively easy analysis of virtualization methods for different scenarios, modulation techniques, and protocols. Additionally, MySVL allows for easy comparison between spectrum- and information-level virtualization, since several information-level protocols exist as OOT modules that can be adapted for virtualization.

We examined the desired isolation functionality for MySVL through simulations, and found that features were operating as intended in an idealized case. However, since the isolation problem is concerned with creating isolated virtual resources, it is necessary to test MySVL in over-the-air experiments, and to examine how spectrum-level isolation can be done under real channel conditions.

7 Spectrum Level Virtualization: The Isolation Problem Part 2

Performing isolation over-the-air

7.1 Introduction

In the previous chapter we developed the The Many-to-many Spectrum Virtualization Layer (MySVL) block for GNU Radio, and verified through simulations that it can achieve spectrum-level isolation under ideal conditions. However, the next step is to test whether MySVL can maintain isolation between virtual wireless links in real world conditions.

It is necessary to do over-the-air experiments for several reasons. Firstly, the isolation problem is about creating independent virtual wireless links, and it is meaningless to propose isolation techniques for virtualization that do not survive real world conditions. By doing over-the-air experiments, we can observe what additional measures are needed for real world virtualization, and where the problems lie. Secondly, we can only analyse the benefits and drawbacks of spectrum-level virtualization through actual operation. Finally, experiments can show us interesting potential applications of spectrum-level virtualization, especially in the context of Fifth Generation Mobile Networks (5G).

Therefore, in this chapter we execute several experiments on the CONNECT test-bed to examine how isolation can be done in practice. Based on our observations, we implement extra measures to enable virtualization at the spectrum level. Finally, we analyse how virtualization performs in several scenarios and propose possible use cases for MySVL.



Figure 7.1: The CONNECT testbed. 16 ceiling-mounted RF front-ends are paired with virtual nodes running software radios. Virtual nodes can be reserved using an automated website and accessed remotely through Secure Shell (SSH).

7.2 Introducing the CONNECT Testbed

The CONNECT testbed at Trinity College Dublin provides a selection of radio frequency front-ends connected to computer nodes, to support investigation into radio communications and networks. The core idea behind the testbed is to enable as much flexibility and reconfigurability as possible for radio research investigation.

The testbed consists of 14 Universal Software Radio Peripheral (USRP) N210 devices [191] and two X310 USRP devices [192], ceiling mounted in a four by four grid as shown in Figure 7.1. These Radio Frequency (RF) front-ends can receive frequencies between 40 MHz and 4 GHz, allowing for a wide range of experiments and flexible spectrum use.

Each radio frequency front-end can be paired with a virtual computer node. These virtual nodes are hosted on a set of servers connected through a private network. The nodes can run different software radios for use with the radio hardware, and since nodes are virtual, many different software radio environments can be provided such as Iris,





Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Reservation system
USRPS' reservation page

Reserve USRP	Start reservation	End Reservation	Frequency	Base Image
☐ <u>13</u>	2017-08-08 09:00:00	2017-08-09 07:00:00	2.40 ▼	GNURadio ▼
☐ <u>14</u>	2017-08-08 09:00:00	2017-08-09 07:00:00	2.40 ▼	Iris ▼
☐ <u>31</u>	2017-08-08 09:00:00	2017-08-09 07:00:00	2.45 ▼	srsLTE ▼
☐ <u>32</u>	2017-08-08 09:00:00	2017-08-09 07:00:00	2.45 ▼	Plain ▼
☐ <u>42</u>	2017-08-08 09:00:00	2017-08-09 07:00:00	Select... ▼	Select... ▼

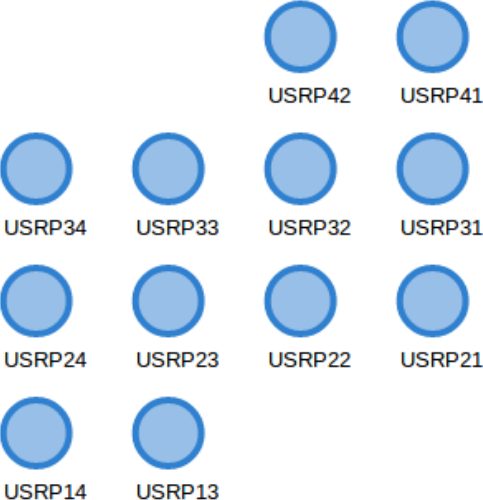


Figure 7.2: The interface of the reservation system. Virtual nodes can be reserved in advance for any period ranging from a few hours to several months. There are several Linux images to choose from, which come with different software radio environments pre-installed.

GNU Radio, SRS Long Term Evolution (LTE) [193], or any custom software radio (see Section 6.4 in the previous chapter for an overview of these software radios).

Virtual nodes and front-ends can be reserved through a fully automated website, shown in Figure 7.2, and once provisioned on the servers, can be used remotely from anywhere. Indeed, the CONNECT testbed is used for teaching radio communication courses throughout the world. Additional information on the testbed can be found at [194] and a tutorial for getting started with the testbed is provided at [195].

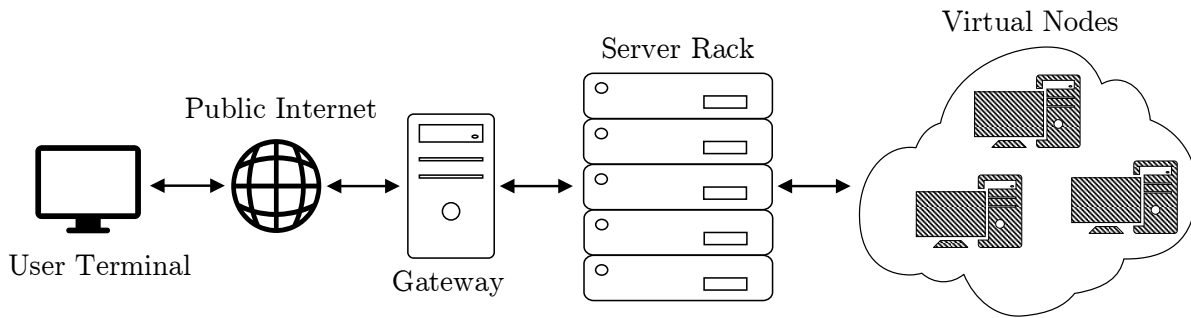


Figure 7.3: A private network was set up, with servers that could host virtual nodes. These nodes were used to run simulations

7.2.1 Involvement in Testbed Creation

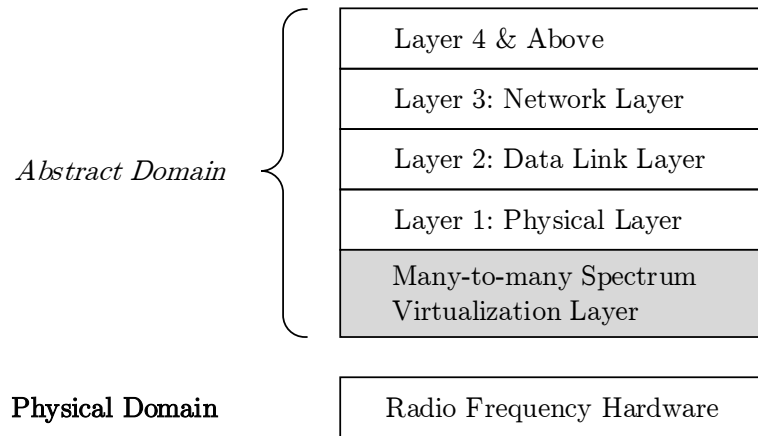
The creation of the CONNECT testbed is the result of a collective effort between many researchers over the course of several years, and the design has gone through several iterations and refinements. Although the testbed is well established now, the first author of this thesis played a part in the earliest design and implementation of the testbed. This work is summarized very briefly below.

Before the CONNECT testbed was equipped with radio frequency hardware, it was necessary to set up the node virtualization environment on the private network with a gateway connecting the network to the outside world, shown in Figure 7.3. Ultimately, the purpose of server virtualization was for use with the testbed, but initially the virtual nodes were used to run simulations. The virtualization application used was OpenNebula [196], an open-source datacenter virtualization framework, which enables the creation of virtual nodes with customizable specifications.

Although the testbed has been improved substantially since then and has many more features, the core functionality and design is still largely influenced by the work that was done several years ago. The role of the first author in the creation of the testbed has been useful in several ways, as it taught him more about virtualization, and also now allows him to benefit from the excellent radio testbed to do experiments on spectrum-level virtualization.

7.3 Experimental Process

As mentioned before, it is necessary to perform over-the-air experiments to analyse isolation successfully. However, we emphasise that the aim of this chapter is not to do a performance analysis on our implementation, but instead to explore how isolation at



Reproduction of Figure 6.2: MySVL sits under the physical layer of the OSI model.

the spectrum-level can be done in practice. Succeeding with this first objective, as a second objective we investigate potential applications of wireless link virtualization at the spectrum level.

Ideally, we would like to keep our implementation of virtualization as simple as possible for several reasons:

Firstly, as MySVL sits below the physical layer in the Open Systems Interconnection (OSI) stack (see Figure 6.2, reproduced above), we would prefer not to reproduce functionality which the Physical Layer (PHY) already does, such as channel estimation, error correction, synchronization, etc. MySVL should be as minimalistic as possible.

Secondly, MySVL needs to be transparent to the PHY, meaning that the PHY should not be aware of any difference when using MySVL compared to the case without MySVL. Adding additional functionality to MySVL can cause overheads which might influence this transparency and cause disturbance. For example, adding a preamble for synchronization purposes will introduce some overhead.

Thirdly, for MySVLs to remain as flexible as possible, we would prefer avoiding the use of specific protocols which might limit the possible methods of isolation, and the potential applications of spectrum level virtualization.

Therefore, the experimental process focuses on two aspects: understanding the minimal effort needed to make isolation work in practice; and investigating how isolation can be done in different ways, such contiguously or non-contiguously, or using different wireless link resource dimensions (frequency, time), and how this affects the performance.

The approach we take is to initially start without any PHY layer features for the virtualization process, and to use the experiments to incrementally explore which PHY features are needed, if any. The benefit of implementing our module through software radio is that we can include only the features we want, and can flexibly switch on functionality as desired to examine different aspects of the design. Additionally, since we employ a commonly used software radio platform, we can reuse existing blocks to perform such features.

In addition to removing as many PHY features as possible, we also do not employ any features of higher layers for the reasons listed above, such as error correction normally done by the Data Link layer.

Thus, we start by establishing the baseline performance of MySVL in Experiment 1.

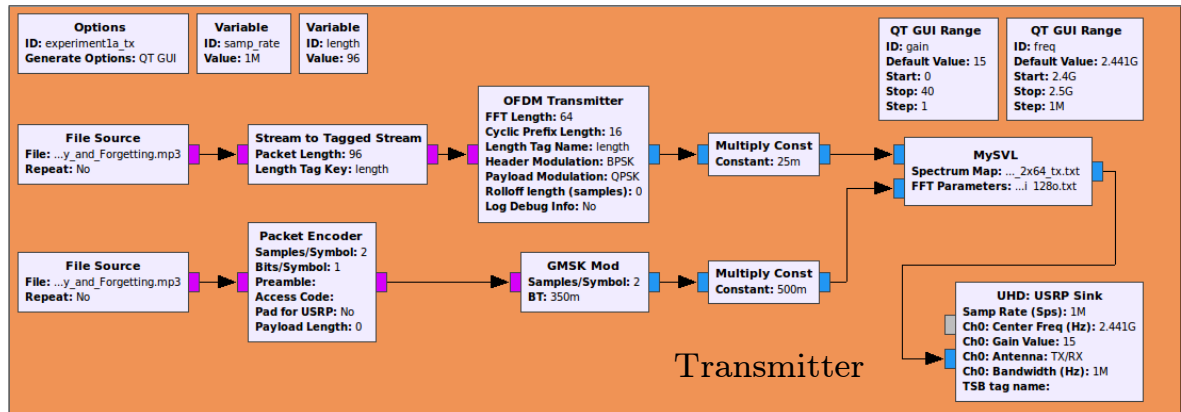
7.4 Experiment 1: Establishing Baseline Performance

The first experiment explores how MySVL performs over-the-air compared to the non-virtualized case, and whether additional measures are needed. We also examine how isolation between virtual resources is affected by the use of different spectrum maps for MySVL. Subsequently, we first examine the scenarios presented in Simulation 1 in Chapter 6, except now split between transmitter and receiver nodes connected to USRP front-ends.

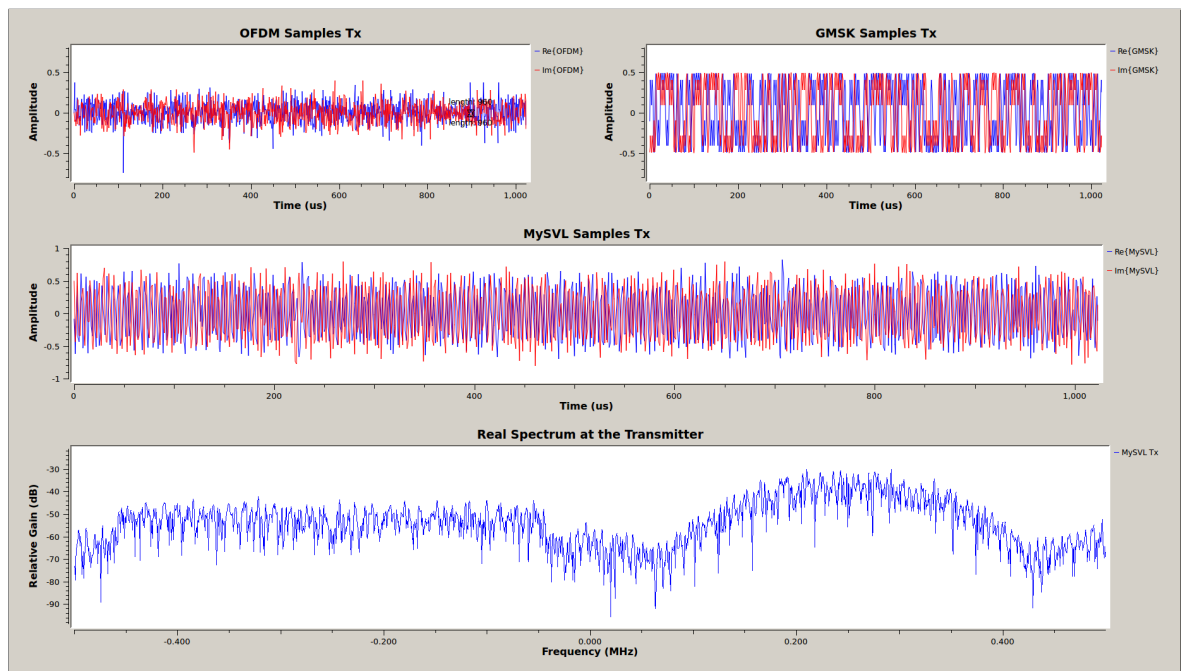
7.4.1 1a) Contiguous Spectrum Map

Our first scenario is based on Simulation 1a in Section 6.6 in Chapter 6. As before, we use two different physical layer modulations to transfer files; an Orthogonal Frequency Division Multiplexing (OFDM) signal, and a Gaussian Minimum Shift Keying (GMSK) signal. Both signals are assigned a virtual bandwidth of 500 kHz, while the real spectrum consists of 1 MHz, and the same contiguous spectrum map is used as in Simulation 1a. We transmit at a frequency of 2.441 GHz and compare the Frame Error Rate (FER) of each signal to the frame error rates of non-virtualized signals.

Note that only very simple PHY signals are used, and we do not perform any higher layer protocols such as error correction or retransmission. Therefore if any error occurs within a packet due to either the MySVL block or due to the channel, the entire packet is dropped.



(a) Transmitter flowgraph for Experiment 1a, based on Simulation 1a. We transmit at a frequency of 2.441 GHz using a USRP front-end.

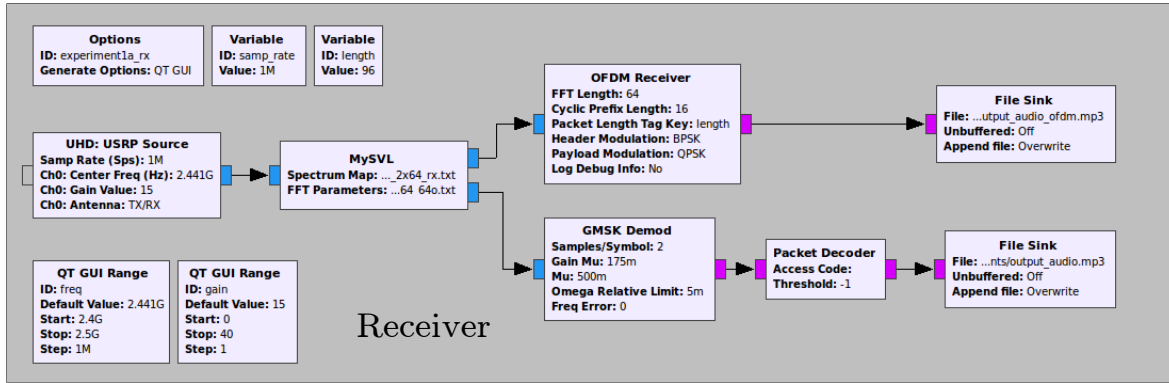


(b) Signal samples at the transmitter shown individually (top row) and after virtualization (middle row), while the bottom row shows the spectrum use.

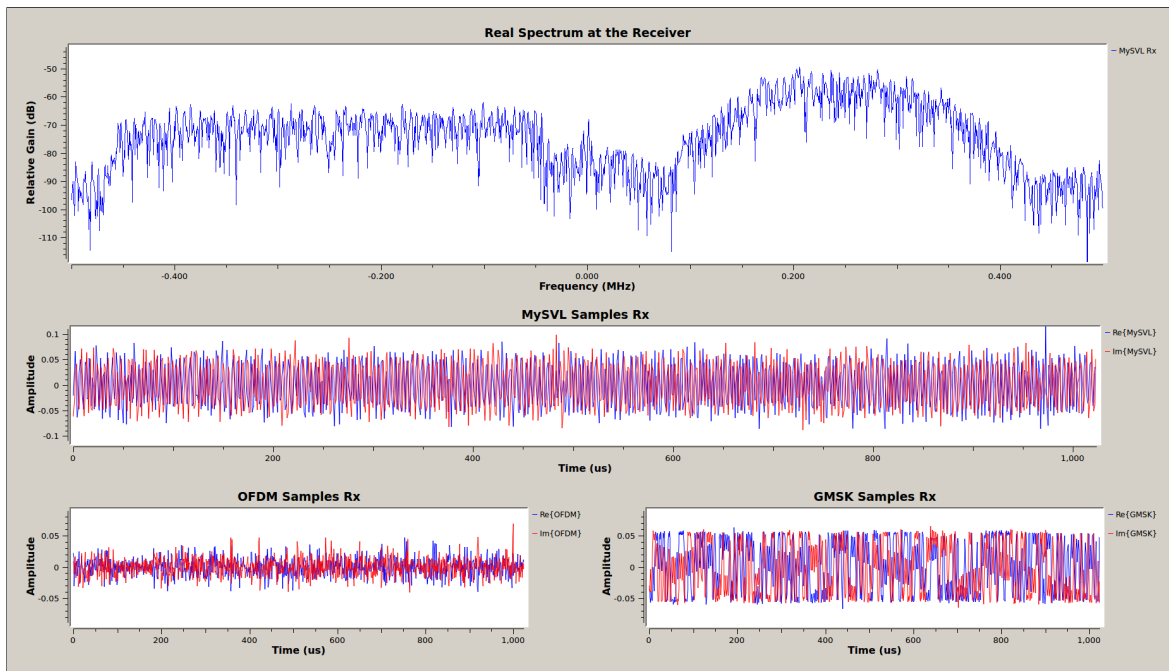
Figure 7.4: Transmitter flowgraph (a) and signals (b) for Experiment 1a. Two different PHY signals are transmitted over the air, an OFDM and a GSMK signal. The spectrum map used in this experiment is the same as the one used in Simulation 1a (Section 6.6 in Chapter 6), as shown by the real spectrum in the bottom graph of (b).

The GNU Radio companion flowgraph is shown in Figure 7.4(a), while Figure 7.4(b) shows the transmitted signal samples and the real spectrum. The corresponding flowgraph and signal samples at the receiver are shown in Figure 7.5

We measure the FER based on the number of correctly received packets. The OFDM signal has a frame error rate of about 4.62%, very similar to the non-virtualized frame error rate of 5.39%. The discrepancy between the two rates might be caused by interference from external signals. However, the GSMK signal barely receives any packets, and



(a) Receiver flowgraph for Experiment 1a, based on Simulation 1a in Chapter 6. We transmit at a frequency of 2.441 GHz using a USRP front-end.



(b) Incoming spectrum use at the receiver shown in the top row, while the corresponding time-domain signal samples are shown in the middle row. The bottom row shows the samples after undoing spectrum virtualization

Figure 7.5: Receiver flowgraph (a) and signals (b) for Experiment 1a. Two different PHY signals are transmitted over the air, an OFDM and a GMSK signal. The spectrum map used in this experiment is the same as the one used in Simulation 1a in the previous chapter, as shown in the bottom graph in (b).

has an FER of 97.5%, whereas the non-virtualized case has an FER of 0.006%!

Although we expected MySVL to have impact on the performance, the only explanation for such a dramatic loss is that the signals are interfering with each other. To test whether this is the case, we repeat the experiment with a lower amplitude for the OFDM signal and a higher amplitude for the GMSK signal. As expected, the FER of the OFDM signal increases significantly to 65.9%, while the FER for the GMSK signal decreases slightly.

Therefore, although the scenario in question worked in simulation, when doing actual transmission over the air, it is necessary to leave additional guard bands between signals to avoid interference. Though this is to be expected, our experimental approach is to start with minimal modifications (compared to the simulated case) and to only adapt the scenario or include new features if absolutely necessary. Therefore we tested the most basic case, showing that guard bands are necessary.

7.4.2 1b) Contiguous Spectrum Map with Guard Bands

After illustrating that it is not possible to use the full bandwidth for each signal without guard bands, we now add filtering and resampling operations to the previous scenario. These blocks are already implemented in GNU Radio, and so we can easily adjust the experiment. The resulting flowgraphs for the transmitter and receiver can be seen in Figures 7.7(a) and 7.6 respectively.

Most importantly, we increase the total bandwidth of the experiment to 3.125 MHz (this number works well with the USRPs), so that we can leave guard bands between the signals. We observe that a guard band of approximately 500 kHz is needed to avoid interference, similar to the guard band used for a 5 MHz signal in LTE [197]. Therefore, we resample the GMSK signal to reduce its bandwidth to a quarter of that of the OFDM signal, which is also a more realistic scenario. We add filtering operations to both signals to reduce the out-of-band leakage. As a result, there is now a guard band of more than 500 kHz between the two signals when sent over the air, shown in Figure 7.7 b). At the receiver, the bandwidth of the GMSK signal is changed back through resampling.

Again, we measure the frame error rate of the two signals. For the OFDM signal, the

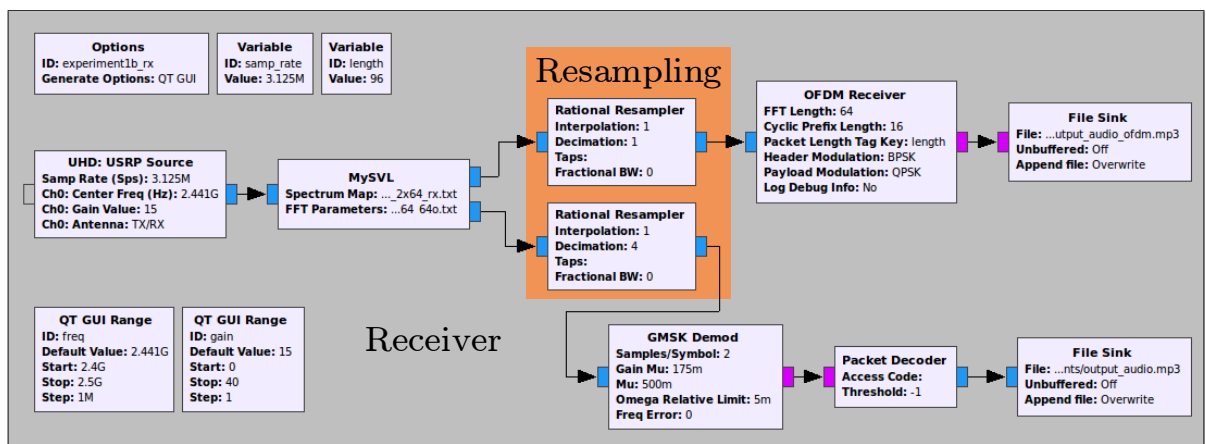
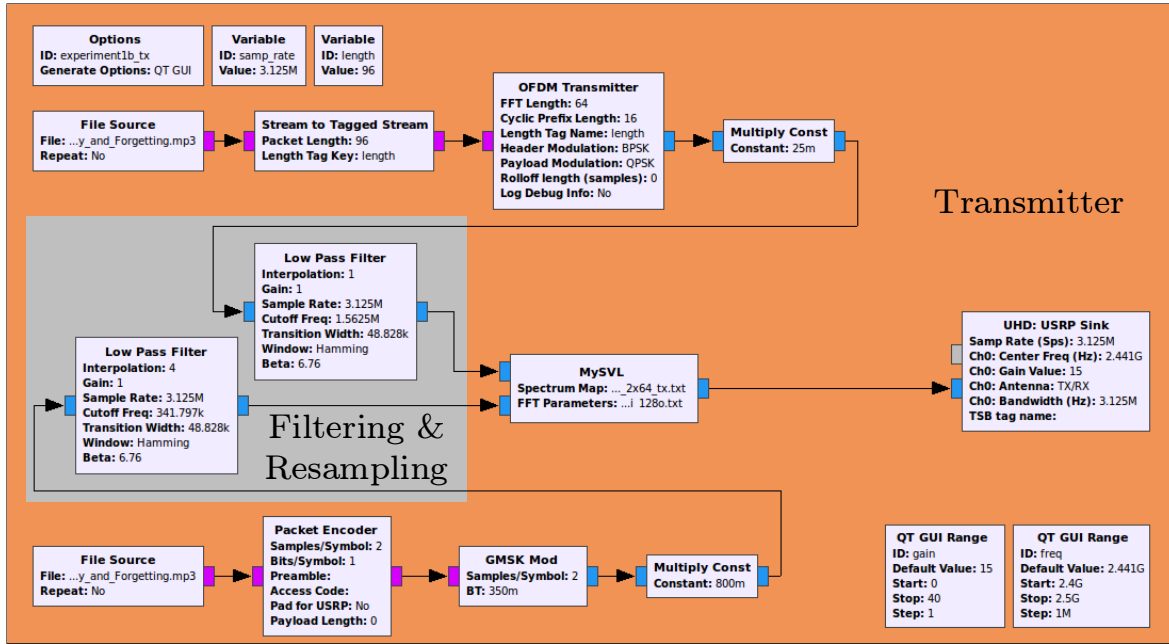
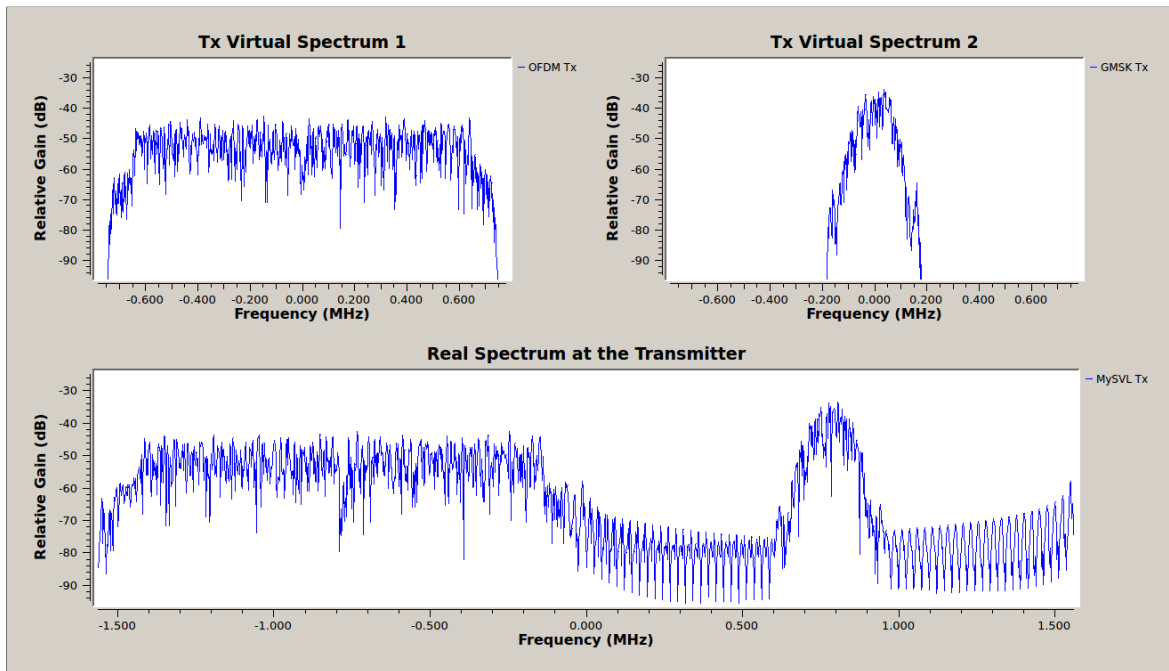


Figure 7.6: Receiver flowgraph for Experiment 1b, showing the resampling operations to change the bandwidth of each signal back to 500 kHz.



(a) Experiment 1b transmitter flowgraph. We use existing GNU Radio blocks to perform filtering and resampling operations to reduce the bandwidth of each signal to 250 kHz.



(b) Virtual and real spectrum graphs at the transmitter for Experiment 1b showing the bandwidth adjustment and the guard band of 250 kHz.

Figure 7.7: Transmitter flowgraph (a) and spectrum graphs (b) for Experiment 1b. Filtering and resampling operations (highlighted in grey) are added to the baseline scenario.

FER remains consistent at approximately 4.75%, while for the GMSK signal the FER is between 0% and 2.75%, depending on channel conditions.

From this experiment, we can conclude that MySVL can achieve isolation between

different virtual links for the contiguous case *without any additional PHY features*, fitting with our perspective of attempting to do as little as possible. However, as is to be expected, sufficient guard bands are necessary to avoid interference (whether from the channel, or from the RF hardware), which we were able to determine for illustrative purposes.

7.4.3 1c) Non-Contiguous Spectrum Map

Next, we are interested in the performance of MySVL over real-world conditions when a non-contiguous spectrum map is used. Although signals might be isolated, the splitting of virtual spectrum might cause additional overheads in comparison to the contiguous case.

We apply the same flowgraphs as used previously in Experiment 1b, with the only exception that the spectrum maps from Simulation 1b are used at the transmitter and receiver. The guard bands between the signals are still approximately 500 kHz, as can be seen in the resulting virtual and real spectrum plots for the transmitter in Figure 7.8

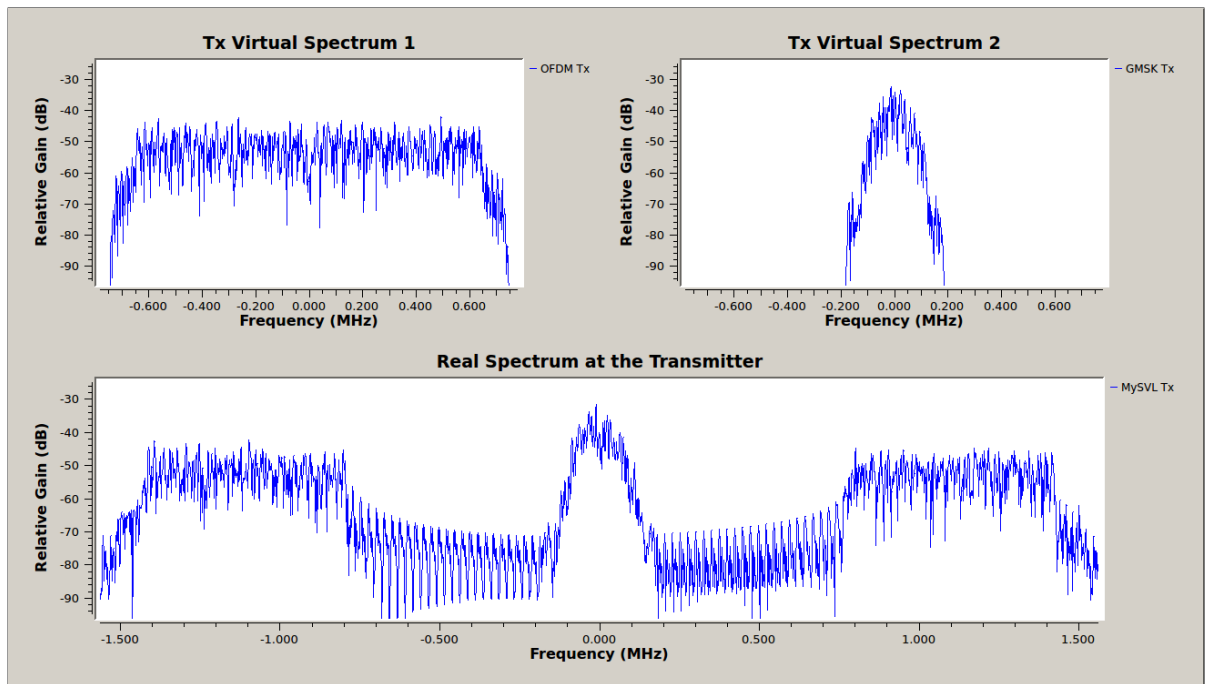


Figure 7.8: Virtual and real spectrum at the transmitter for the non-contiguous case (Experiment 1c).

When we measure the frame error rates of the two signals, we discover that the splitting of virtual spectrum introduces significant issues. Although the FER of the GMSK signal remains the same as in Experiment 1b, the FER of the OFDM signal varies

wildly between about 57% to 98%, meaning that sometimes we receive almost no packets, while other times we receive approximately 43% correctly.

Therefore, it is not possible to use non-contiguous spectrum maps with only the basic MySVL block, and we need to consider what might be causing such a large error rate in real-world conditions.

7.4.4 Observations

From the above experiments we found that MySVL can be used as is for contiguous spectrum maps, so long as the signals are not directly beside each other. However, non-contiguous spectrum mapping results in a very large performance loss.

One of the reasons for the large FER in the non-contiguous case is the misalignment of the Fast Fourier Transform (FFT) window at the receiver with the corresponding time-domain samples, as pointed out by the authors of the Spectrum Virtualization Layer (SVL) paper [124]. Additionally, the large fluctuation in the FER suggests that sometimes the FFT windows are more synchronized, while at other times the FFT windows are less synchronized.

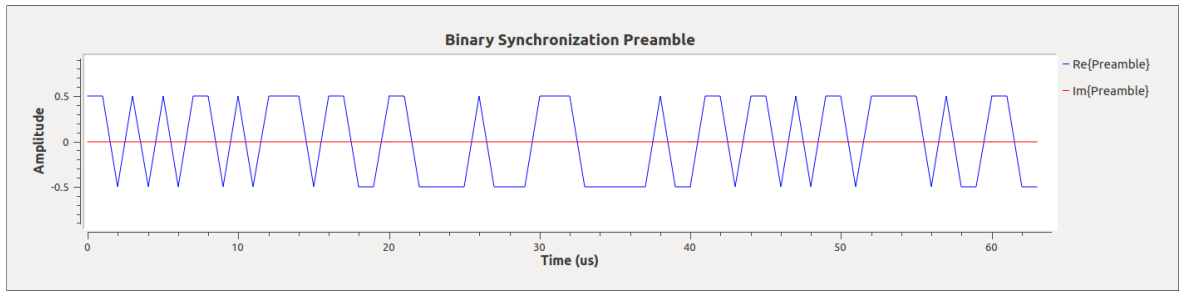
Therefore, in the next section we introduce means of synchronizing the receiver and aligning the FFT window to the correct time-domain samples.

7.5 Frame Synchronization

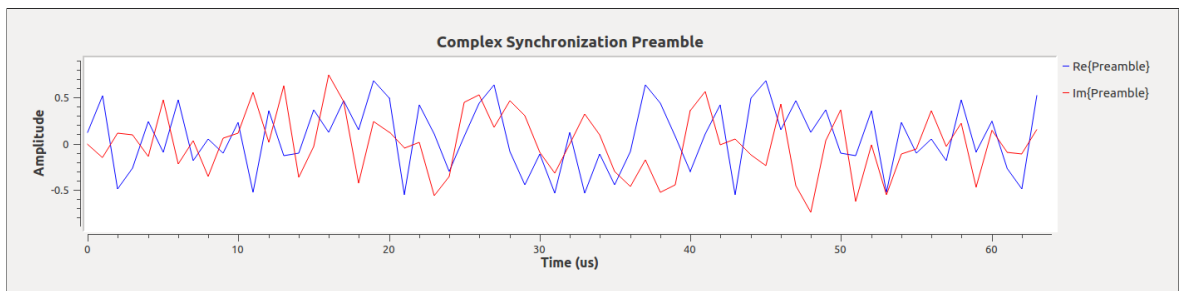
At the receiver, an FFT is used on the time-domain samples to convert the signal to the frequency domain. However, if the FFT window is not aligned correctly to the time-domain samples, it could cause samples to be mapped wrongly in the frequency domain. As such, it is necessary to synchronize the receiver so that the FFT window is aligned correctly. We call a series of FFT windows a frame, as essentially we want to achieve frame synchronization.

Although there are many means of achieving frame synchronization at the receiver, we need to use a synchronization method that relies on a unique preamble because our implementation of wireless link virtualization is generic. The reason behind this design choice is that if we use a synchronization method that does not use a unique preamble, then we could potentially synchronize incorrectly if a signal is transmitted that uses the same synchronization technique. For example, if we use autocorrelation to synchronize

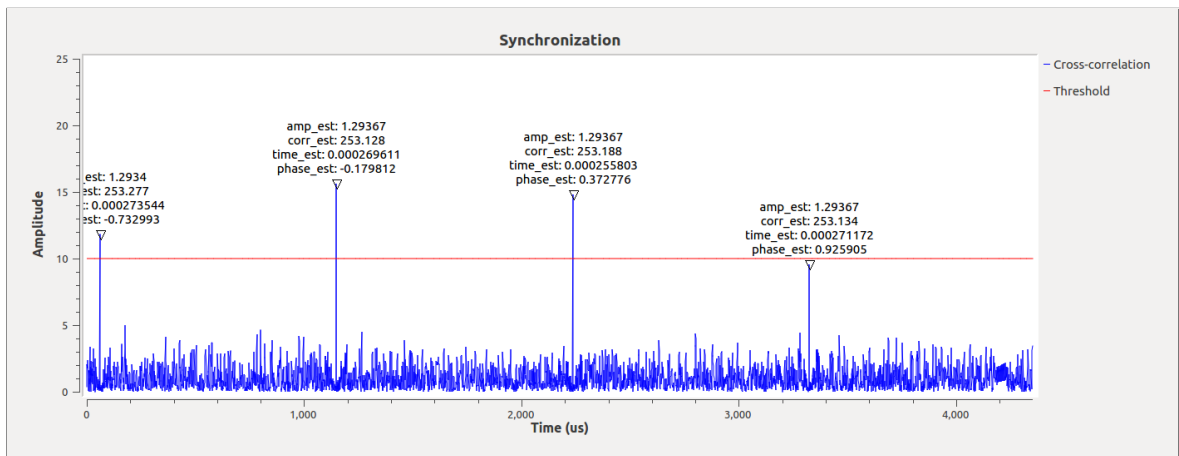
in time, then we might get false positive synchronizations when transmitting an OFDM signal, as OFDM also uses auto-correlation for synchronization [198].



(a) 64 bit sequence with low aperiodic autocorrelation properties, similar to Barker codes.



(b) Complex Hermitian sequence with good autocorrelation properties for synchronization, generated by taking the FFT of (a).



(c) Preamble detection at the receiver. Incoming samples are cross-correlated with sequence (b), shown in blue. A large spike above a threshold indicates that the preamble sequence has been found. The correlation estimator block also adds GNU Radio tags with time and phase offset estimates whenever it detects the preamble. Note that the cross-correlation spikes vary in height due to the carrier frequency offset.

Figure 7.9: Frame synchronization is done by adding a 64-bit preamble from [199] with low aperiodic autocorrelation properties at the transmitter (a). To make the signal more robust to noise we take the FFT of this signal, shown in (b). At the receiver this sequence is detected reliably using cross-correlation and thresholding (c), which enables us to know where frames start.

Therefore, we use cross-correlation estimation block which is already implemented in GNU Radio. The cross-correlation estimation block requires a known time-domain

sequence, and constantly performs cross-correlation on the incoming samples. Using a unique known sequence allows us to be sure that we are synchronizing at the right time.

There are many possible sequences we could choose to use for synchronization, depending on the properties which we desire. We choose to use a 64-bit binary sequence with low aperiodic autocorrelation [199], shown in Figure 7.9(a). This sequence has similar properties to the properties of Barker codes [199]. However, if this binary sequence is not robust enough to channel effects, taking the FFT of this real binary sequence creates a Hermitian sequence of complex samples (Figure 7.9(b)) which also provides very good autocorrelation properties. We tested both sequences and found that the autocorrelation properties were very similar, and thus we use the complex sequence for synchronization.

When the cross-correlation block detects the preamble in the incoming stream, it uses GNU Radio stream tags [200] to indicate the start of a frame and additionally estimates the phase offset and the time offset. However, we found that cross-correlation detection is not very accurate and additionally relies on somewhat arbitrary thresholds. Instead of using the tags provided by the cross-correlation block we use the absolute value of the cross-correlated output. By setting a threshold, we reliably detect the start of an FFT frame, as shown in Figure 7.9(c).

One observation to note is that synchronization does not need to occur at every frame. In fact, frame synchronization is mainly needed at the start of the transmission to line up the received samples to the FFT window of MySVL. However, after some time, it might be necessary to resynchronize because of the time offset.

7.5.1 Developing a Triggered Demultiplex Block

Although we can now detect the start of a frame reliably, there is no block in the base installation of GNU Radio which we can use to remove the preamble from our stream of complex samples at the receiver. In essence, we need a demultiplex block that can separate complex samples into multiple streams. Surprisingly, this block does not exist and so it is necessary to develop it.

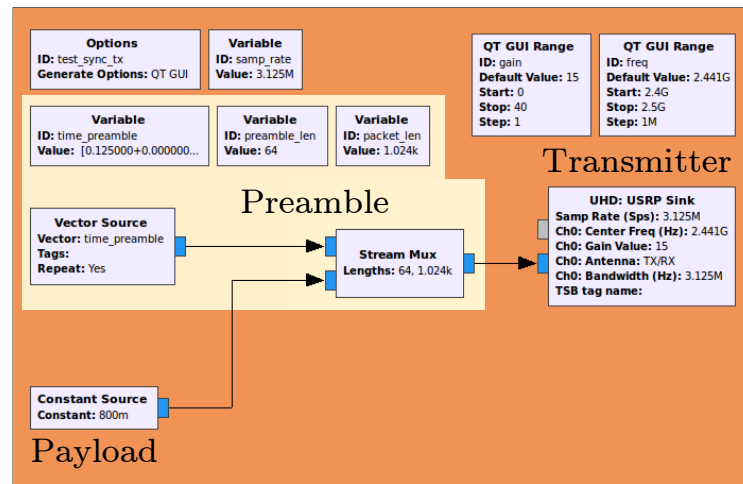
In addition to separating the preamble samples from the input samples to MySVL, it should be possible to reset the demultiplex block so that we can synchronize when the preamble is detected. Therefore, the block should have some sort of trigger or reset function, and the option of setting a delay after the trigger occurs.

We develop a triggered demultiplex block for GNU Radio with the desired functionality. As a design choice, we choose to drop any frames that are not of the right length,

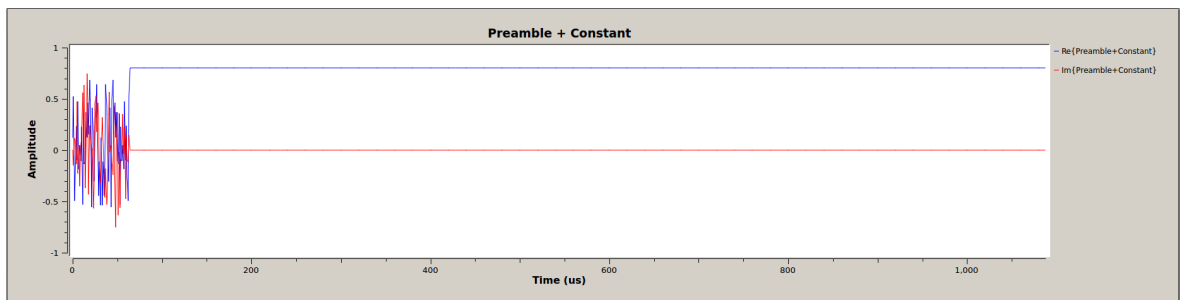
as these frames would cause problems for MySVL. Additionally, it is necessary to add GNU Radio stream tags to the first sample of every frame.

Although we mentioned GNU Radio tags before, we did not explain what there are yet. Stream tags are a means of passing metadata about a stream *synchronously* between blocks [200]. A tag containing information, such as the length of a frame, is added to a specific sample in the signal, and this information travels with the sample through the rest of the flowgraph, where it can be read by downstream blocks.

The triggered block adds GNU Radio stream tags at the start of every frame indicating the length of the following frame. We modified the MySVL block to synchronize to these tags, ensuring that the FFT windows are correctly line up.



(a) Transmitter flowgraph multiplexing a synchronization preamble with a constant signal.



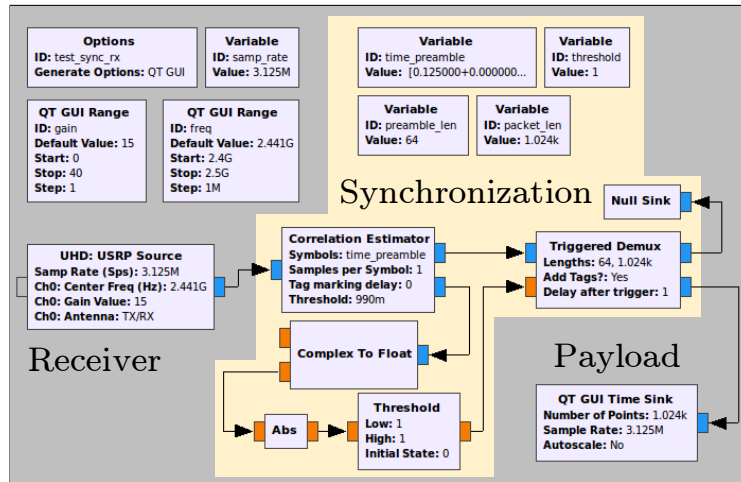
(b) The resulting signal, with the preamble at the start followed by the payload signal, in this case a constant signal.

Figure 7.10: Flowgraph at the transmitter (a) and the preamble appended to the payload samples (b).

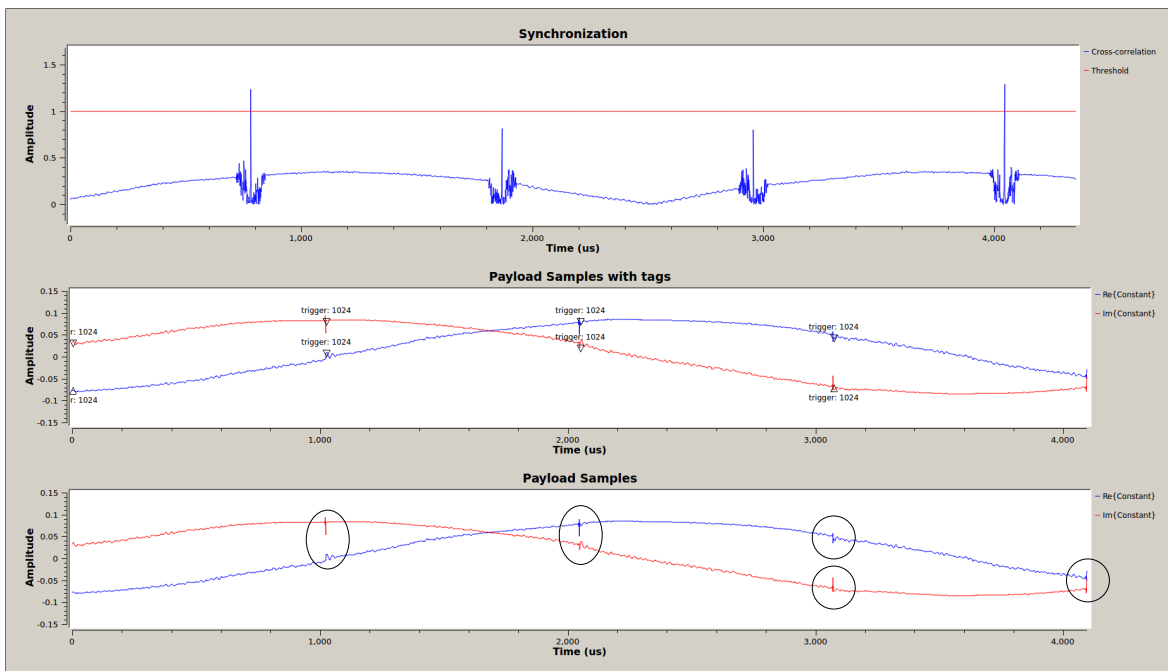
7.5.2 Testing Synchronization on the Testbed

We test how synchronization performs on the testbed using a very simple setup shown in Figures 7.10 and 7.11. At the transmitter, we send the preamble and a constant signal

to examine how this signal will be affected. We use a simple multiplex block to insert the preamble into the stream of samples.



(a) Synchronization at the receiver. Incoming samples are cross-correlated with the known preamble, and the absolute value of the output is thresholded to create a binary trigger signal. The triggered demultiplex block separates the preamble from the payload and adds tags to show the start of every frame. Whenever the trigger signal goes high, the demultiplex block resets.

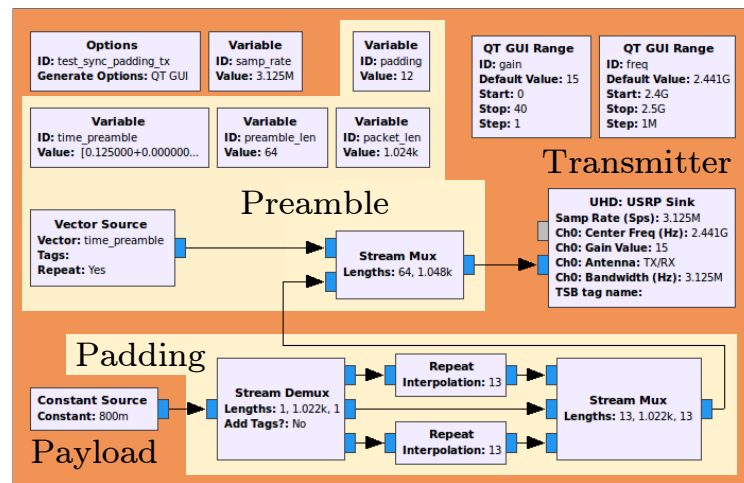


(b) The top graph shows the preamble detection as before, while the bottom graph shows the resulting output. While the preamble has been removed and tags have been added for synchronization, samples at the start and end of each frame have been distorted, highlighted with the black circles.

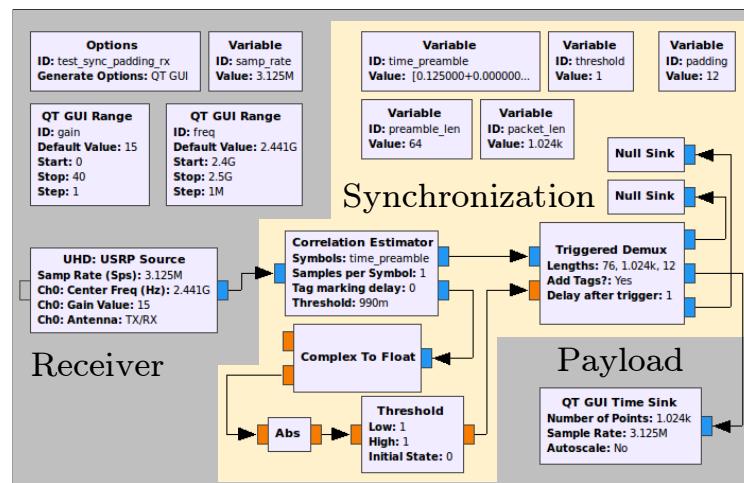
Figure 7.11: Receiver flowgraph for synchronization (a), and the preamble detection and removal shown in (b). However, the payload frames in the bottom graph of (b) display reoccurring artefacts at the start and end of each frame.

At the receiver, the incoming samples are passed to the correlation estimation block, and the preamble is detected using thresholding. The resulting output of the threshold block is a binary trigger which is zero except where a preamble is detected. The trigger signal is passed into the triggered demultiplex block we developed. This block also receives the unmodified incoming stream of samples from the correlation block. When the trigger is activated, the triggered demultiplex block resets and passes the next n_1 complex samples to output stream 1, as defined in the parameters. Subsequently, the following n_2 complex samples are passed to output stream 2. In this way we can separate the preamble from the payload (in this case the constant signal).

Figure 7.11 b) shows the resulting payload samples. Although, the preamble has been removed successfully, there are several distortions to point out. Firstly, the signal has changed from a constant signal to a sine wave as a result of the Carrier Frequency Offset (CFO). Secondly, several artefacts have been introduced at the transition between



(a) Transmitter flowgraph for synchronization with padding. The first and last sample of each frame are repeated as padding samples between the preamble and the payload.



(b) At the receiver the padding is removed using the triggered demultiplex block.

Figure 7.12: The flowgraphs for the synchronization test with padding.

frames. From close observation, it is possible to see that the cause of this is due to the Gibbs phenomenon [201] occurring somewhere in the hardware chain.

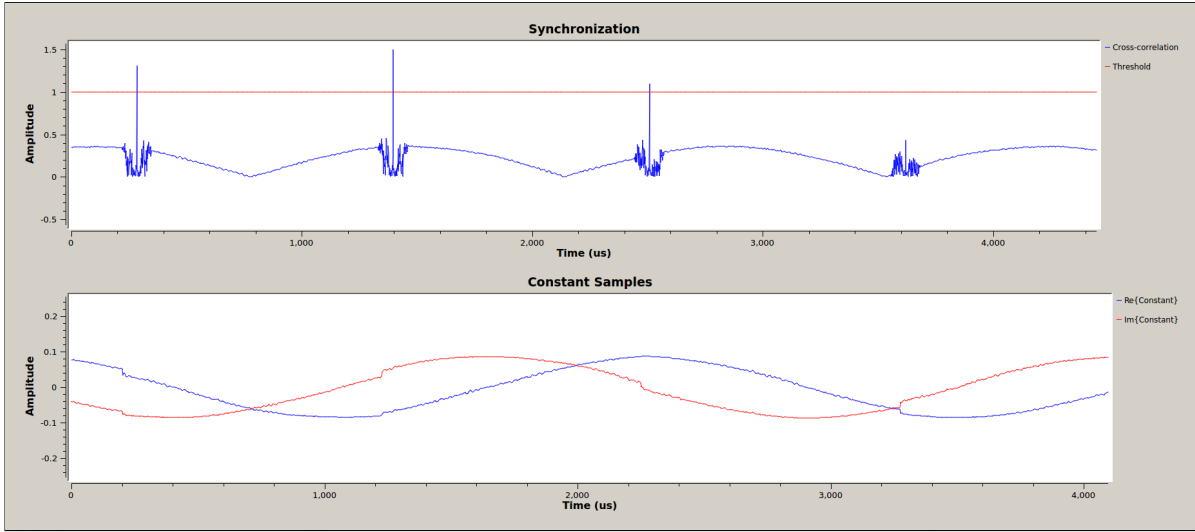


Figure 7.13: Through the use of padding, the artefacts caused by the Gibbs phenomenon can be avoided. However, a small jump between samples occurs between frames, caused by the frequency offset.

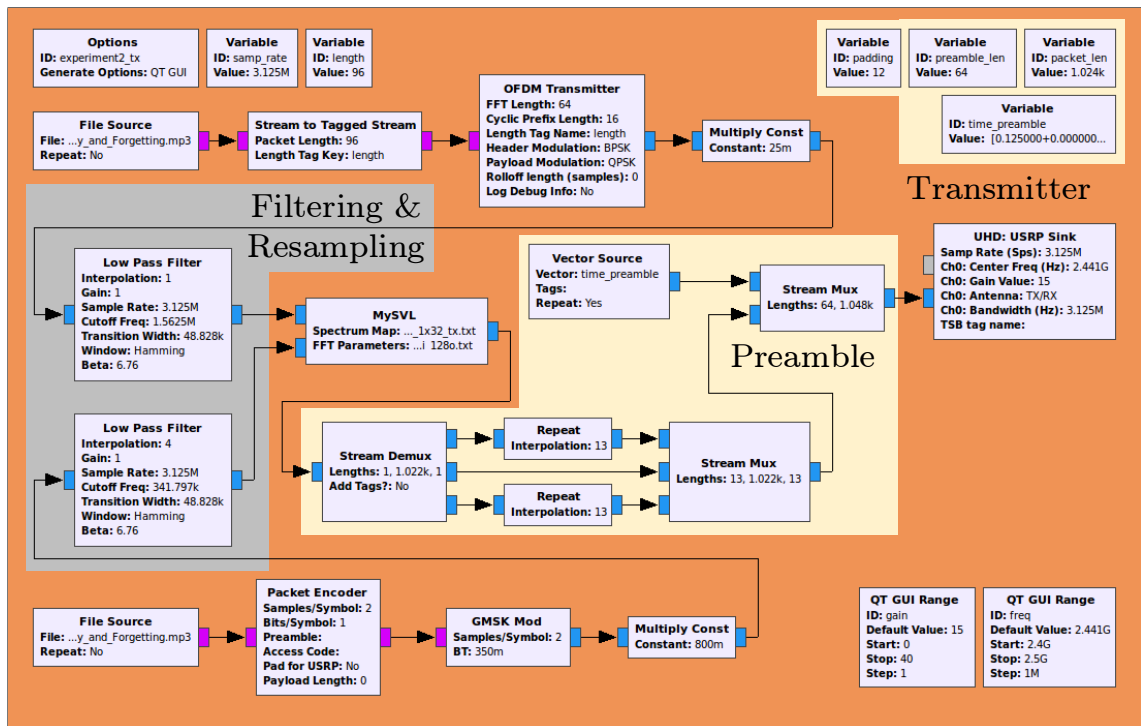
7.5.3 Removing the Artefacts

These artefacts can cause significant problems to a signal as they introduce unwanted discontinuities and errors. Therefore, we use padding before and after the preamble, by repeating the first and last sample of the payload. This padding performs a similar role to the cyclic prefix in OFDM, except that we do not use it for channel estimation.

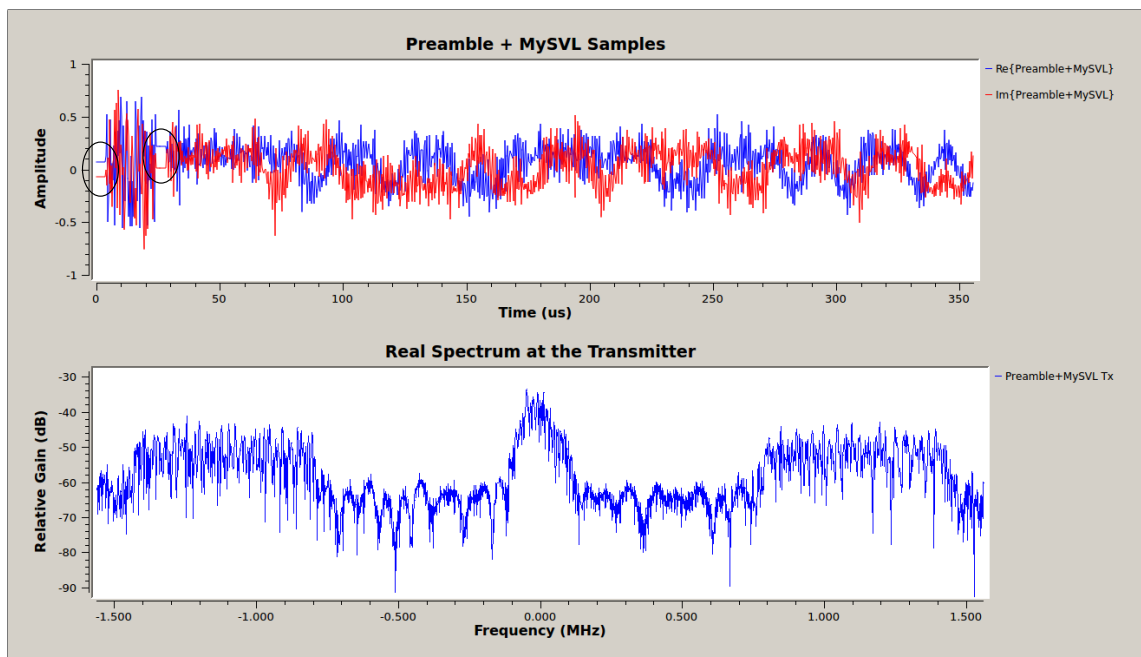
We repeat our synchronization test on the testbed with the addition of padding. Figure 7.12 shows the modifications to the flowgraphs at the transmitter and receiver, while the output payload samples are shown in Figure 7.13. We can see that adding a small prefix and suffix has resulted in the elimination of the artefacts. However, a small discontinuity remains due to the carrier frequency offset.

7.6 Experiment 2: Adding Frame Synchronization to MySVL

Having developed a means of accomplishing frame synchronization, we now test how frame synchronization increases the performance of MySVL in the non-contiguous case.

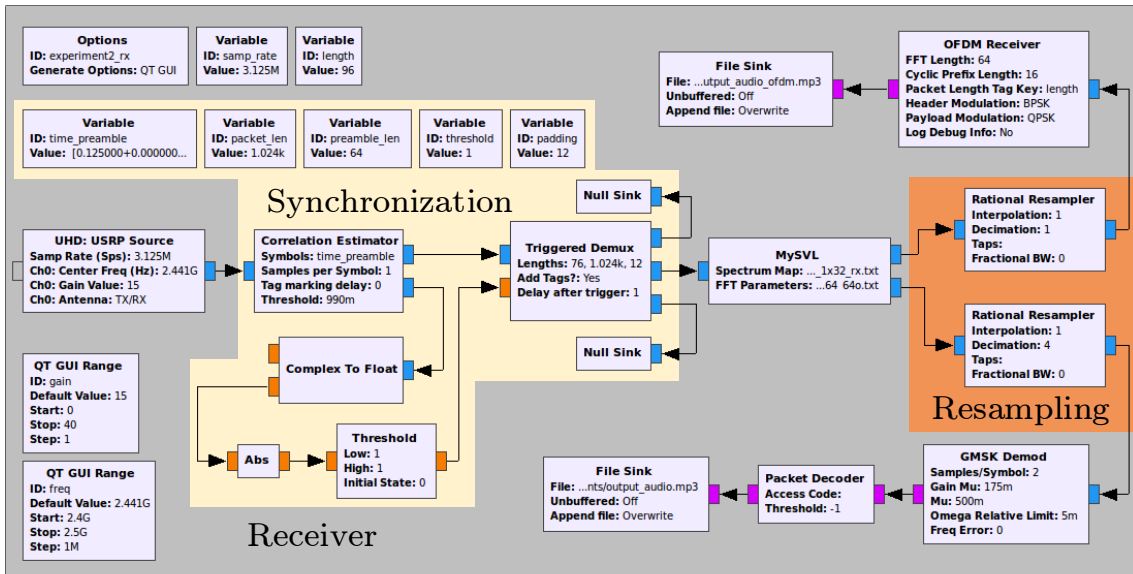


(a) Flowgraph for Experiment 2 transmitter. We build on Experiment 1c (non-contiguous spectrum mapping) by adding a preamble for synchronization and padding to overcome the limitations of the hardware.

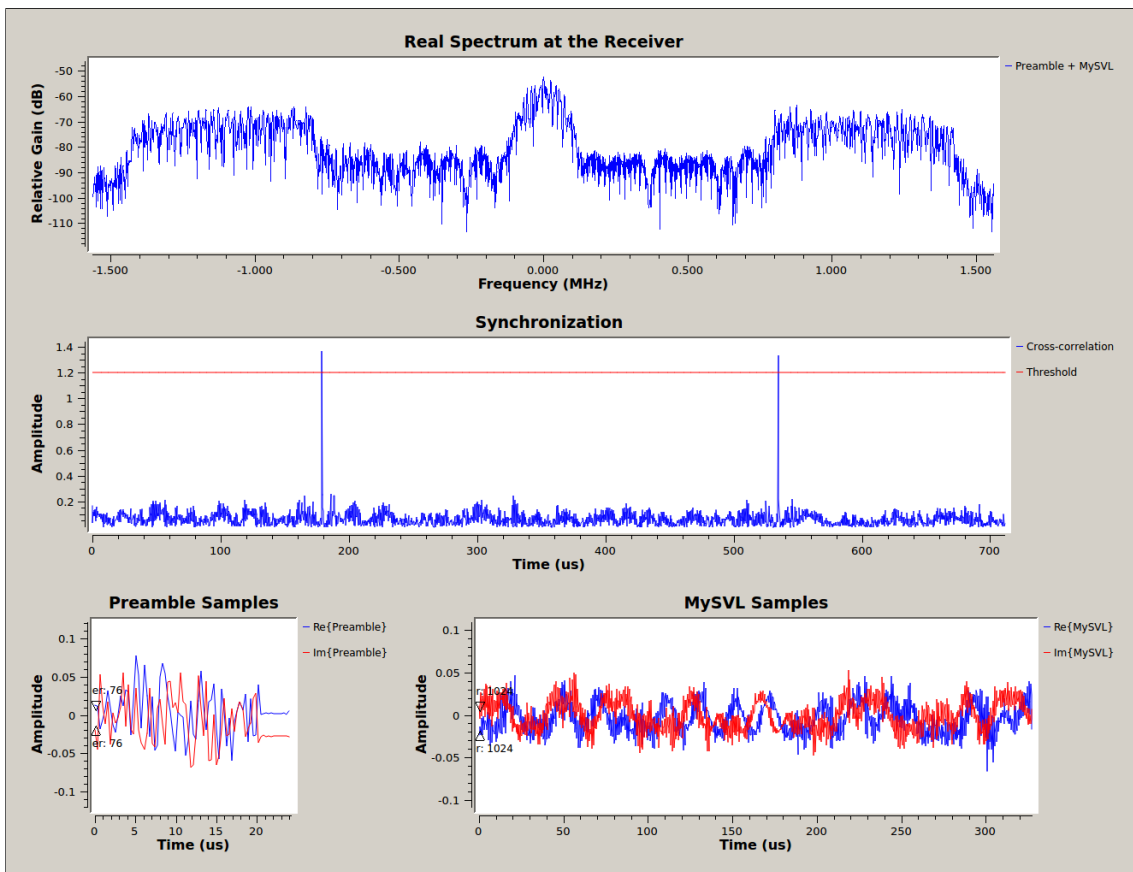


(b) The top graph shows the composition of a transmitted frame. Padding precedes and follows the synchronization preamble, after which come the payload MySVL samples. Padding is highlighted by the black circles. The bottom graph shows the spectrum, which now changes periodically due to the preamble.

Figure 7.14: Experiment 2 analyses the performance of MySVL when frame synchronization is employed. The flowgraph for the transmitter is shown in (a), while the transmitted samples and spectrum are shown in (b). Note that the spectrum now changes because of the synchronization preamble.



(a) Flowgraph for Experiment 2 receiver. Synchronization is done by cross-correlating incoming samples with a known preamble. The triggered demux block separates the payload samples and ensures that the samples going to the MySVL block are aligned with the FFT window.



(b) The real spectrum received is shown in the top graph, while the middle graph shows the preamble detection through cross-correlation. The graphs on the bottom row show the separation of the preamble from the payload samples.

Figure 7.15: Examining the effect of synchronization on MySVL performance in the non-contiguous case. Synchronization ensures that the FFT window aligns to the correct time-domain samples.

We test the same scenario as in Experiment 1c, i.e. using MySVL to allocate an OFDM signal and a GMSK signal to non-contiguous spectrum. However, in addition we perform frame synchronization as described. Figure 7.14 shows the flowgraph at the transmitter and how the preamble is multiplexed with the signal. The corresponding flowgraph at the receiver is found in 7.15(a).

We measure the frame error rate by comparing the number of bytes transmitted to the number of bytes received correctly for several different packet sizes (or frame lengths). The frame error rates for the OFDM and the GMSK signals are shown in Figure 7.16. As we can see from this plot, the GMSK signal has an FER of approximately 3 - 3.5%, which is slightly worse than in the case without synchronization, but still acceptable. One reason for the deterioration is that the use of the preamble and the padding has resulted in a discontinuity caused by the frequency offset, causing some errors did not occur previously.

The OFDM frame error rate is now around 12%, a substantial improvement on the previous best case FER of 57%. Synchronization has massively increased the number of packets received in the non-contiguous case, and moreover allows the non-contiguous case to be used reliably, as previously the FER fluctuated hugely.

7.6.1 Observations

By adding synchronization, we have mitigated most of the errors caused by the channel when doing spectrum virtualization non-contiguously. The number of packets received correctly by the OFDM signal has increased by a large amount.

However, there is still a significant difference between the non-contiguous case and the contiguous case. We believe that in the dropped frames errors are only occurring in several bits and that the bit error rate is low. However, with no easy method of measuring the bit error rate, we can only assume this.

Since it is very difficult to measure the bit error rate directly, we must use alternative methods to investigate the effect of the channel on spectrum virtualization. One such method is to transmit a constant signal of some amplitude through MySVL on one stream, and a zero signal on the other stream. Then we can examine the output of the two streams individually at the receiver and inspect how the samples are affected. When we use such a method, we observe the output samples shown in Figure 7.17.

In this graph, we can see that MySVL causes some errors to occur, as there are “notches” occurring at regular intervals of 64 samples (the notches are almost too small

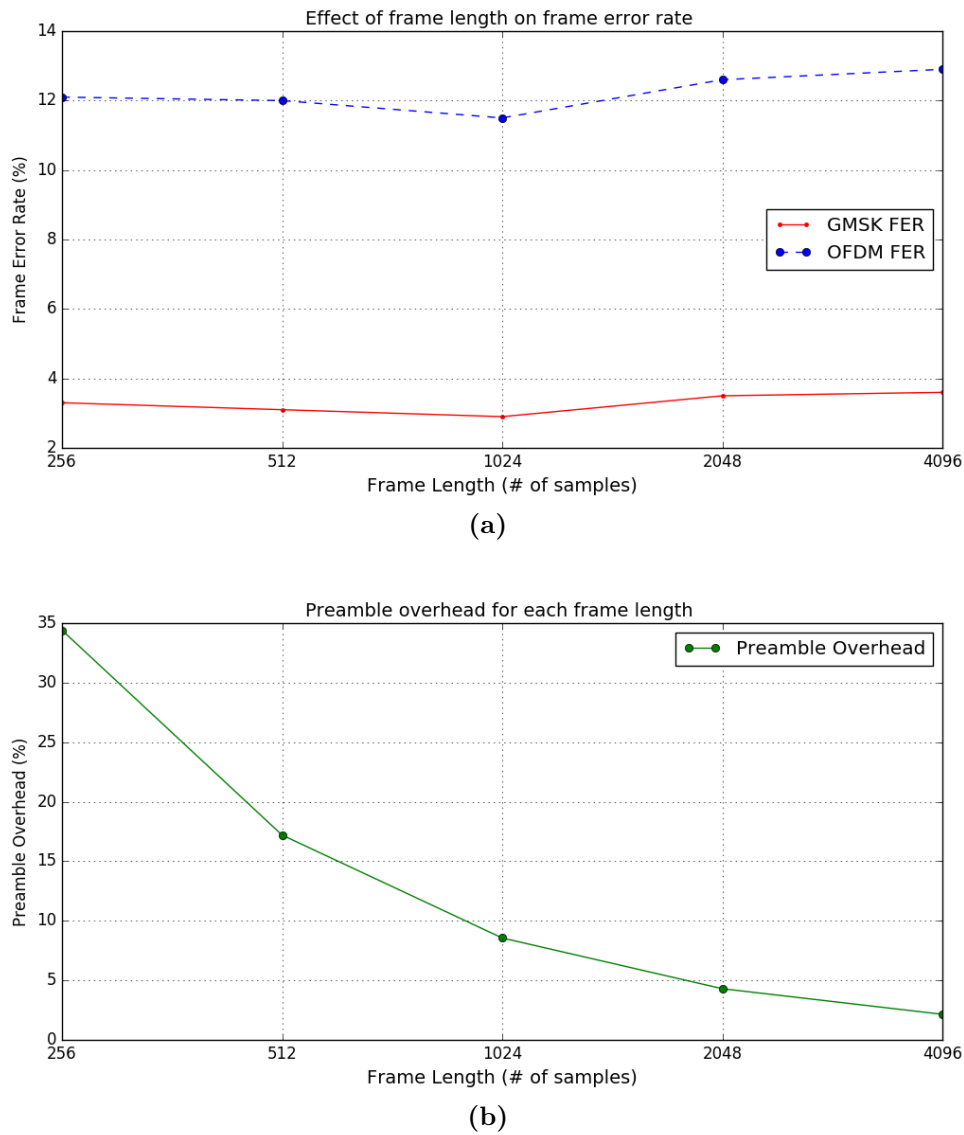


Figure 7.16: Frame error rates for OFDM and GMSK signals for different frame lengths (a) and the preamble overhead per frame length (b). Although the FER does not change much, the overhead decreases significantly as the frame length increases.

to see in the top graph). Since the FFT for each stream is 64 samples, these errors are occurring at the start and end of each FFT window.

Because we know that there is some discrepancy caused by the carrier frequency offset between frames (this is shown in Figure 7.13), we believe that the CFO is causing MySVL to map some samples incorrectly in the frequency domain. Therefore, we now focus on correcting for the carrier frequency offset, and investigate how this affects performance.

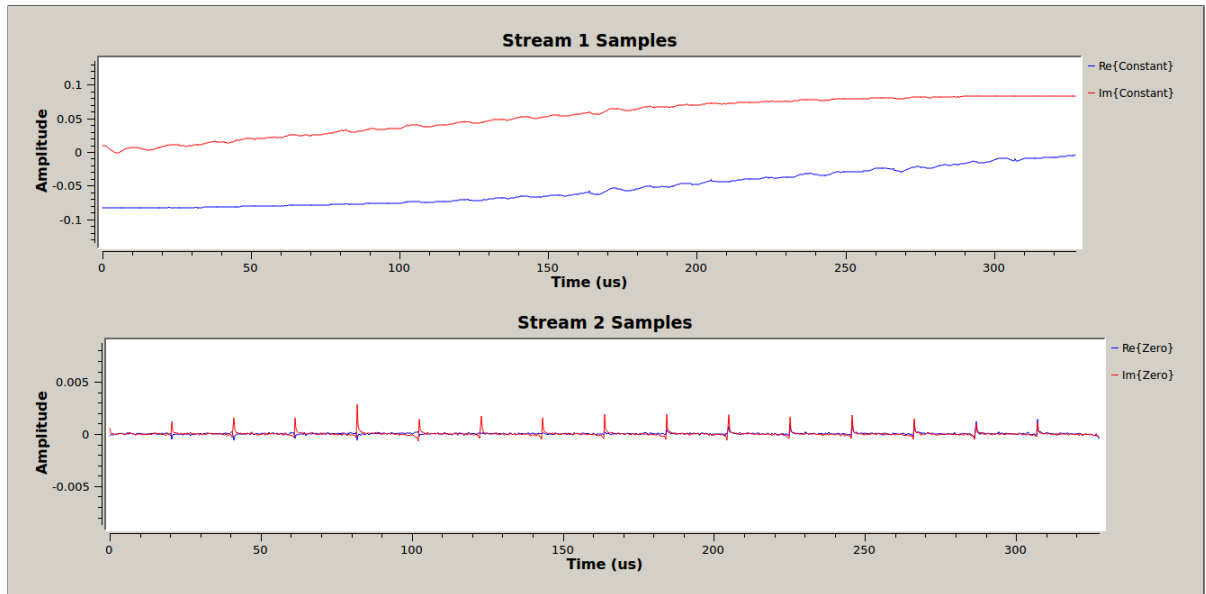


Figure 7.17: MySVL output samples at the receiver for a constant stream and a zero stream. From the graphs we can see that “notches” are occurring at regular intervals—the FFT window size. We think these notches are caused by the carrier frequency offset which causes samples to be mapped wrongly in the frequency domain.

7.7 Carrier Frequency Offset Estimation and Correction

There are many methods of estimating and correcting the carrier frequency offset. Broadly speaking, there are two main groups of techniques that are commonly used: phase-locked loop techniques for narrowband signals, surveyed in [202]; and for wide-band signals, techniques using preamble sequences. An excellent survey of synchronization techniques for Orthogonal Frequency Division Multiple Access (OFDMA) is given in [203], while techniques for estimating the CFO are discussed in [204].

Several carrier frequency offset estimation and correction methods are implemented in GNU Radio already, such as the Costas loop technique for phase-modulated narrowband signals [205], and Schmidl and Cox (S&C) synchronization for OFDM signals [198]. Since we are not using a narrowband signal, we decide to use the existing Schmidl and Cox block to perform CFO estimation.

The Schmidl and Cox block consists of two steps, frame detection followed by frequency offset estimation. Reference sequences with repetitive properties are used for both of these stages, as described in detail in [203]. However, frame detection is not very reliable, since instead of detecting a clean spike, a peak of several samples is used. Therefore, we only use the S&C method for CFO estimation, which is done by comparing two identical halves of the reference sequence, which remain identical after channel conditions except for a phase shift caused by the frequency offset.

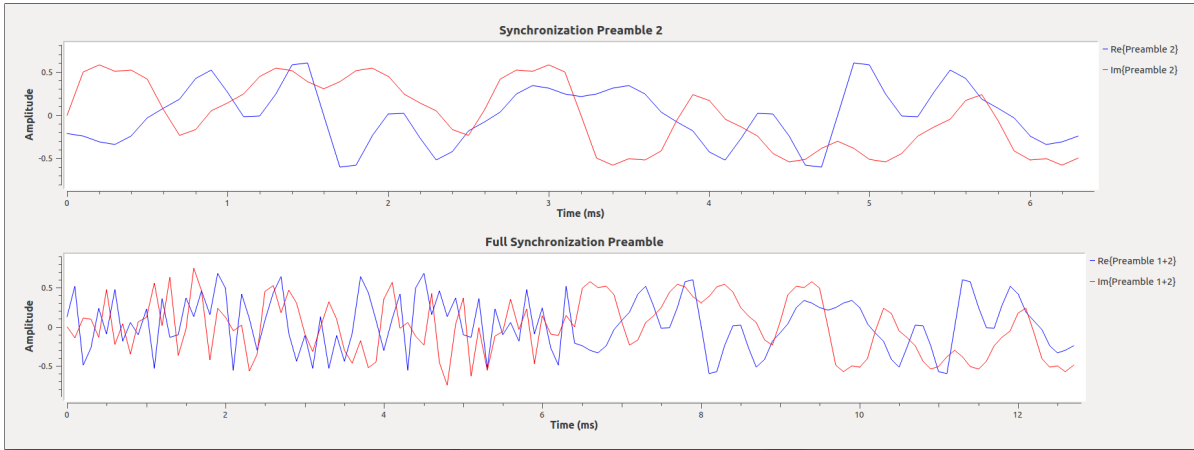


Figure 7.18: The top graph shows the second preamble used for frequency offset estimation, while the full preamble sequence is displayed in the bottom graph.

Our existing preamble sequence can be used with the S&C block in GNU Radio for rough frequency offset estimation, as it is a Hermitian sequence. We add a second preamble sequence, generated by taking the Inverse Fast Fourier Transform (iFFT) of a real sequence with zeroes for odd indices, which provides fine frequency offset calculation [198]. The second preamble sequence and the combined synchronization preamble is shown in Figure 7.18. However, the frequency offset estimation is only done periodically at lengths of 64, which means that the resulting output has large “jumps”. To correct for this, we use a moving average filter which smoothes the frequency offset estimation.

Having estimated the carrier frequency offset as ε , we can correct the phase offset by multiplying every sample by $e^{j\pi\varepsilon}$ using existing GNU Radio blocks.

7.7.1 Testing CFO Estimation and Correction on the Testbed

We test the carrier frequency offset estimation and correction techniques on the testbed, using a constant signal as when testing frame detection. The flowgraph at the transmitter is the same as the flowgraph previously shown in Figure 7.12(a) except that the preamble now consists of the two 64-length sequences. The flowgraph at the receiver is shown in Figure 7.19(a).

Figure 7.19(b) shows the results of the frequency offset estimation and correction. Firstly, frame synchronization is performed as previously, and the preamble samples are separated from the payload. The preamble samples are used as input to the Schmidl and Cox block. We only pass in our preamble samples to the block in case there is a similar preamble in the payload samples, which might cause the S&C block to estimate the CFO incorrectly.

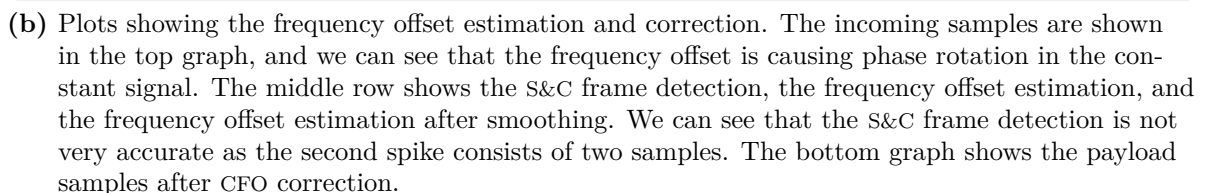
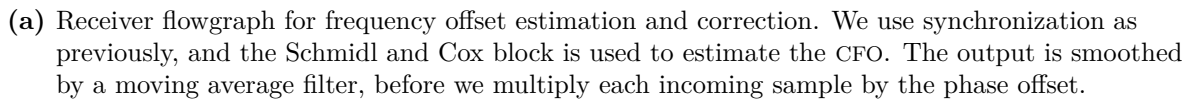


Figure 7.19: Flowgraph (a) and plots (b) for CFO estimation and correction at the receiver.

Next, the CFO estimate is smoothed using a moving average filter of length 2048. We tested several lengths and found that this length was most suitable, although it did not change the results much. The output of the moving average block is used to correct the frequency offset for the incoming samples. It is necessary to use all of the incoming samples (both preamble and payload) when correcting for the CFO, and to separate the preamble from the payload again *after* CFO correction because otherwise the “jumps” between payload frames (shown previously in Figure 7.13) would not be removed correctly.

As shown in the bottom graph of Figure 7.19(b), the CFO estimation and correction improves the signal significantly as it reduces the phase rotation and removes a large part of the discontinuities between frames.

7.8 Experiment 3: MySVL with CFO Estimation and Correction

Now that we have developed a means of estimating the carrier frequency offset and correcting the phase rotation it causes, we apply this to spectrum virtualization. The scenario we test is the same as previously – non-contiguous spectrum isolation for an OFDM signal and contiguous spectrum for a GMSK signal. We use the two synchronization preambles at the transmitter, and the flowgraph is essentially the same as the flowgraph for experiment 2 except that we use a packet length of 4096; see Figure 7.14(a). The flowgraph for the receiver now includes CFO estimation and correction, shown in Figure 7.20.

As before we measure the throughput correctly received in bytes. We find that the frame error rate of the GMSK has increased slightly from about 3.5% to 4.9%, while the FER of the OFDM signal decreases slightly from 12% to 10.7%.

7.8.1 Observations

Although CFO estimation and correction provides a very minor improvement, overall all it is not worth doing in our case as it adds a lot of additional complexity, and moreover increases the preamble length, thus increasing the overhead caused by the virtualization process. Although we did not measure the complexity directly, we used the computational requirements as a proxy for measuring the complexity. When using

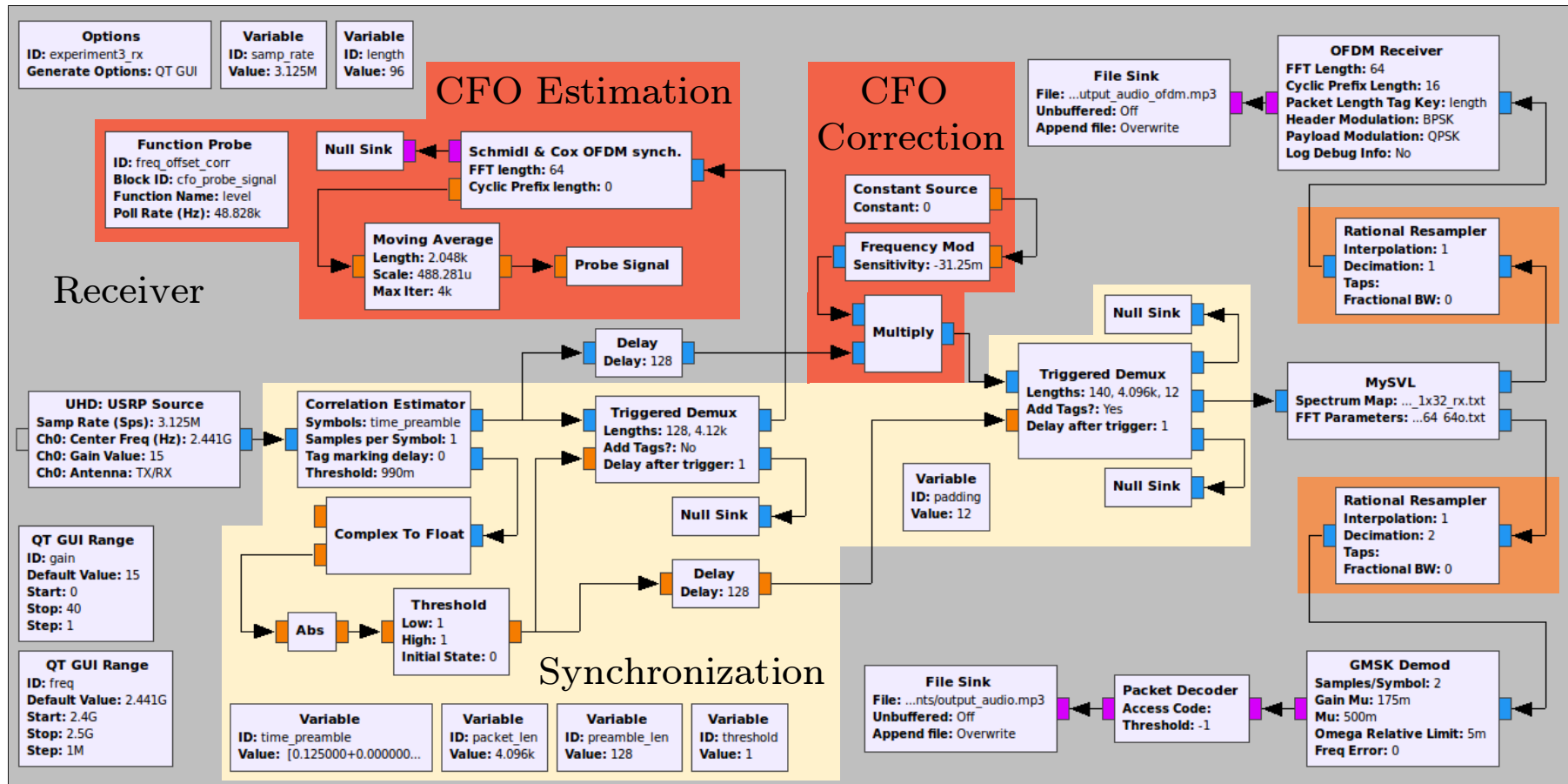


Figure 7.20: Receiver flowgraph for MySVL with carrier frequency offset correction. Synchronization is done as before, and preamble is used by the S&C block to estimate the CFO. The received samples are phase shifted by the estimated CFO value to recover the original signal as closely as possible.

CFO correction, we started to get significant underruns, which means that the computer cannot generate samples fast enough to keep up with the RF front-end [206]. Therefore, we set the sample rate at 3.25 Mega samples per second (Msps) based on this complexity. Although previous experiments were initially performed with higher sampling rates (indicating that they are less complex to run), we reran the experiments with a sampling rate of 3.25 Msps to do a fair comparison. Ultimately, it was useful to investigate whether CFO estimation and correction is useful, even if the performance gains do not justify its inclusion.

As we mentioned briefly before, for the sake of simplicity and transparency we do not employ any features from higher layers, such as Forward Error Correction (FEC), in these experiments. However, the use of FEC would mitigate the effect of real-world conditions and correct for the errors introduced by the CFO, decreasing the FER of non-contiguous spectrum maps.

7.9 Potential Applications

Having examined how spectrum-level isolation can be achieved over-the-air, in this section we propose several potential 5G applications for MySVL. MySVL can be used to do many-to-many RF front-end virtualization and many-to-many wireless link virtualization at the spectrum level. As discussed in Chapter 4, RF front-end virtualization allows RF front-ends to be shared or combined by independent wireless signals. Wireless link virtualization enables the link resources to be shared or aggregated independently, meaning that different technologies and configurations can be used.

There are many potential applications of RF front-end virtualization and wireless link virtualization at the spectrum level, and we discuss several below.

7.9.1 Multiple Virtual Links per Real Link (many-to-one)

Wireless link virtualization and RF front-end virtualization allow wireless links to be shared between multiple virtual links, which we demonstrated in the above experiments. One situation in which this is useful is the case of a wireless node using multiple applications and services simultaneously. Using wireless link virtualization and RF front-end virtualization allows each service to have its own virtual wireless link, tailored to its requirements.

There are many benefits of many-to-one virtualization, such as:

1. Customizability is possible per service, meaning that each link can use its own technology and protocols.
2. New functionality can occur, for example it might be possible to do “live link migration” to move a virtual link from one real link to another link with minimal interruption to the user of the virtual link. Similarly, there can be many other potential new services that are possible because of wireless link virtualization.
3. More efficient spectrum use, as virtual links can be used dynamically and flexibly only when needed in respond to different services demands. Multiple services can share a real wireless link, reducing the need for a dedicated wireless link each.
4. New wireless waveforms, technologies, and protocols can be tested in real conditions alongside existing services, speeding up the development and adoption of new technologies.

7.9.2 Combining Real Links into a Virtual Link (one-to-many)

In addition to splitting real links, virtualization also enables real links to be combined into virtual links with a higher aggregated throughput. We briefly examined whether it was possible to achieve one-to-many virtualization on one of the X310 USRPs on the CONNECT testbed which have multiple transmitter and receiver antennas. Although we have not yet been able to run this experiment correctly, we hope to investigate this further in the near future.

One-to-many wireless link virtualization can provide several benefits compared to existing techniques. Firstly, a higher throughput can be offered to a virtual link than is possible on any real link, *without deploying new RF equipment*. This means existing equipment can be reused, even when new protocols or technologies are developed, which can be very valuable in an industry such as wireless communications, where new generations of technologies are deployed regularly.

Another advantage of one-to-many wireless link virtualization is that links can be made more reliable, since if one of the real links is affected by significant interference or noise or, even worse, by equipment failure, then the remaining real links can still be used for communication. This could be an important aspect for critical Internet of Things (IoT) communications such as self-driving cars.

7.9.3 Additional Applications

As with any virtualization technology, there are many applications and possibilities that are not known in advance, but which could be discovered as link virtualization is used. For example, many-to-one and one-to-many wireless link virtualization can both be used as many-to-many virtualization for new purposes that can not be predicted.

A second development which wireless link virtualization could enable is the convergence of wireless network technologies, meaning that a variety of air interfaces, protocols, and frequency bands could be seamlessly integrated [207].

7.9.4 Flexible Spectrum Use and Dynamic Spectrum Access

In addition to many-to-many applications, virtualization can be used for flexible spectrum use and Dynamic Spectrum Access (DSA). Virtualization enables resources to easily be reconfigured, allowing technologies to quickly be changed. Additionally, resources can be assigned to different users dynamically, enabling DSA.

There is already a significant body of work in the area of DSA and spectrum sharing (see the surveys [188–190]), and wireless link virtualization can be seen as a potential enabler for DSA. Additionally, virtualization can operate alongside or independently of DSA techniques, e.g. either users of virtual resources could use/allow spectrum sharing, or virtualization could be applied to spectrum which is shared. However, although DSA appears to be similar to Wireless Link Virtualization (WLV), it is important to note that there are significant differences. Firstly, the users of virtual wireless resources are unaware that they are sharing the spectrum or accessing the spectrum dynamically and they do not experience any interference from other users if isolation between virtual resources is achieved correctly. Secondly, users of virtual spectrum resources do not need any special DSA techniques, such as spectrum sensing algorithms, control channels or connections to databases, and/or scheduling methods such as game theory auctions or other techniques. Lastly, there is no need for the distinction between different types of users such as primary or secondary users in WLV, although it is possible to use these frameworks if desired. In summary, DSA and WLV are complementary technologies, and WLV can be used as a means of performing DSA, but it is not a requirement.

One example application of flexible spectrum use is the use of wireless link virtualization and RF front-end virtualization in an LTE network shared between commercial operators and a Public Safety (PS) operator, which is discussed later in Section 8.6.

To show how it is possible to use DSA with virtualization, we now describe the use

of spectrum splitting by a Secondary User (SU) to avoid a Primary User (PU). We demonstrate this application on the testbed, described below.

Demonstration 1: Splitting Spectrum to Avoid a PU

One potential application which we can demonstrate already is the use of wireless link virtualization to perform dynamic spectrum access in order to avoid a primary user. In this experiment, a PU is transmitting a GMSK signal in the center of a channel. Spectrum virtualization in the form of splitting spectrum and using a null signal allows a SU to transmit an OFDM signal simultaneously while avoiding any interference with the PU. The setup for the demonstration is shown in Figure 7.21.

The secondary user transmits a null signal in the center of the bandwidth, and virtualization is used to split an OFDM signal so that it uses the outer frequencies. We use frame synchronization as previously with a frame length of 4096 samples, but we do not perform any CFO estimation and correction. However, the preamble we use for synchronization has a flat spectrum occupying the entire bandwidth, and when we multiplex this preamble with the MySVL samples the real spectrum at the transmitter periodically occupies the entire bandwidth as shown in Figure 7.22(a).

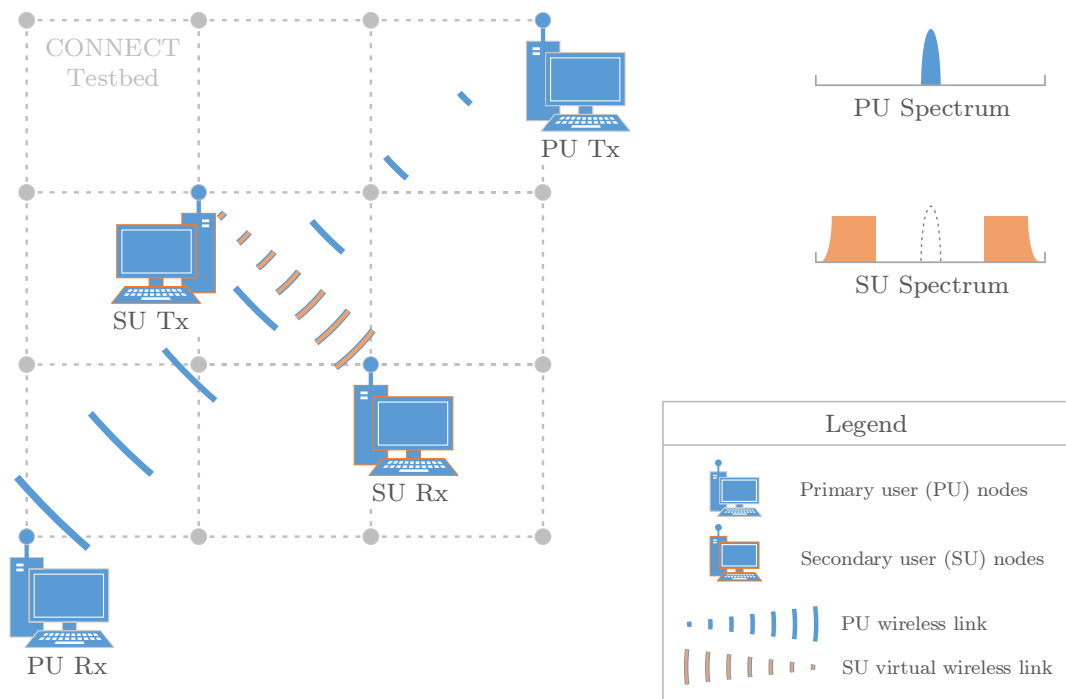
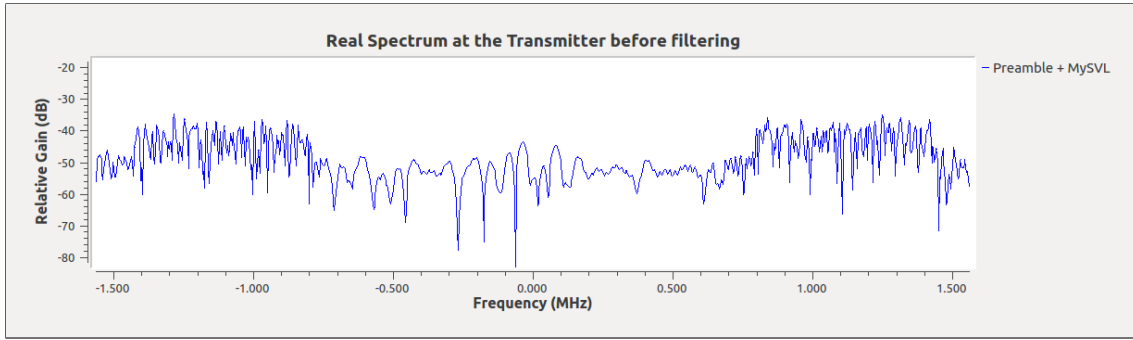
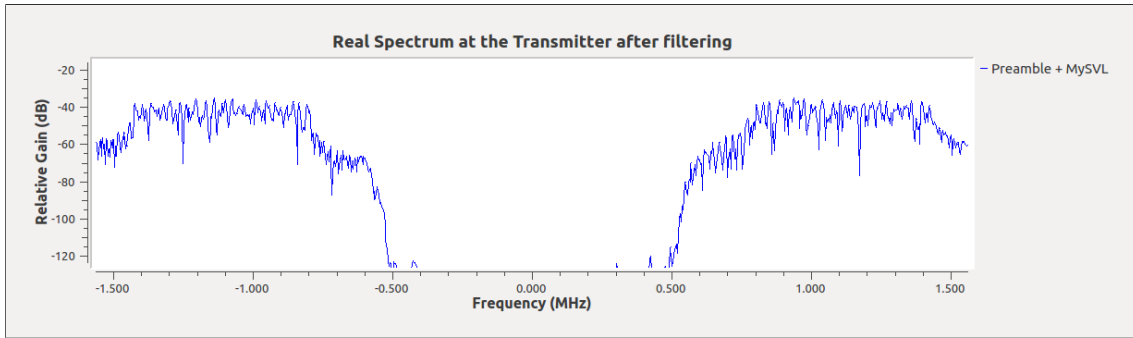


Figure 7.21: A primary user is transmitting a narrowband signal in the center of a channel from the two furthest nodes in the grid. Through the use of wireless link virtualization in the form of splitting, a secondary user can dynamically access the unused real spectrum and transmit an OFDM signal simultaneously, without affecting the PU



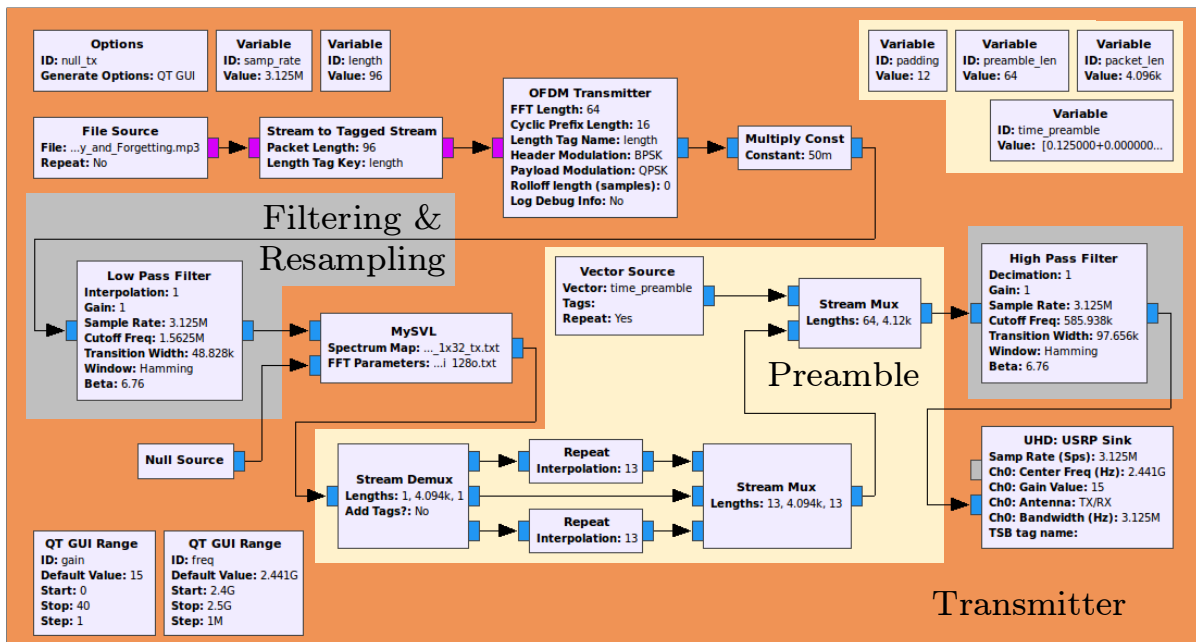
(a) Real spectrum at the transmitter before filtering.



(b) Real spectrum at the transmitter after high-pass filtering.

Figure 7.22: Spectrum at the Tx before (a) and after (b) suppressing the central frequencies.

To create a signal which has a null in the middle, it is necessary to reduce the power of the center frequencies. Therefore, we add a high pass filter to the transmitter flowgraph (Figure 7.23) which suppresses the central frequencies as shown in Figure 7.22(b).

**Figure 7.23:** SU transmitter flowgraph with the filtered preamble.

We also experimented with creating a new synchronization preamble which only occupies the outer frequencies. However we found that simply filtering the existing preamble gave better results.

The real and virtual spectrum at the secondary user receiver is shown in Figure 7.24. We can see that the PU signal exists in the center of the channel, and that it would not be possible for the SU to use any part of the channel without virtualization, since it would interfere with the PU. By splitting the spectrum however, the SU can maintain a large enough guard band to avoid interfering with the PU.

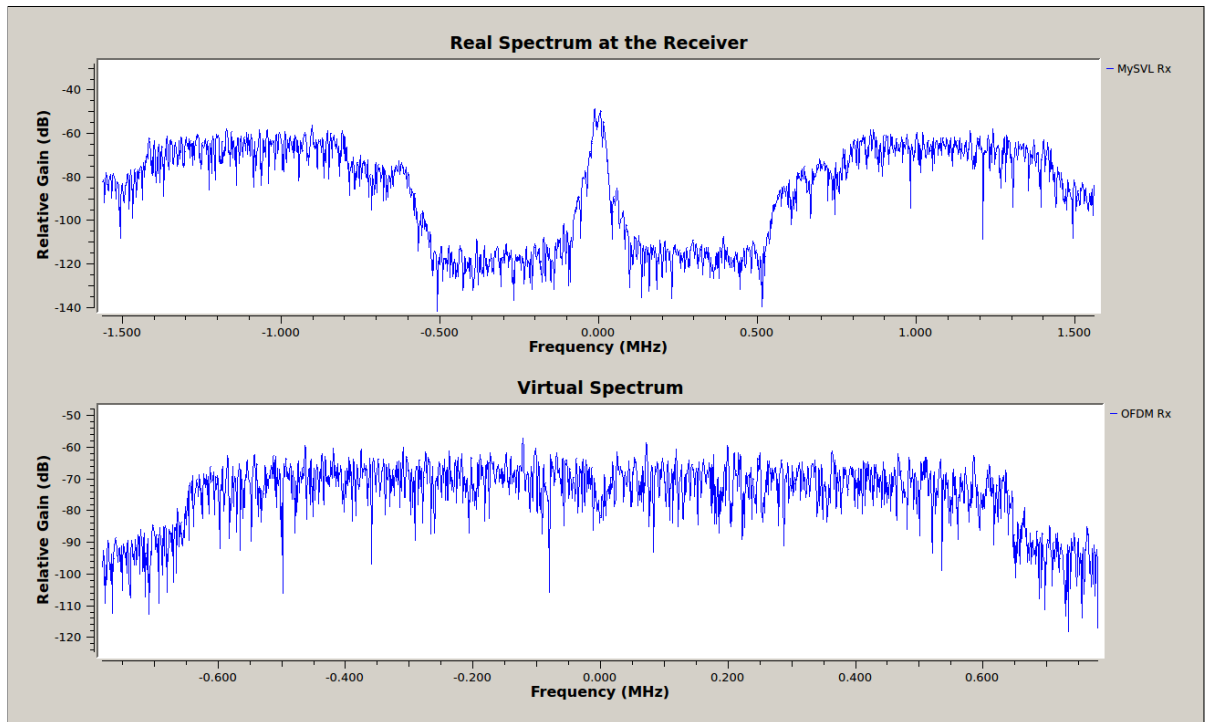


Figure 7.24: Real and virtual spectrum at the SU receiver.

We measured the frame error rate of the PU and found that it was 0.02%, which shows that the SU did not, or barely, interfere with the PU.

For the SU, we first measured the FER without the PU transmission to establish the effect of the filtered preamble, and then measured the FER with the PU transmission. We found that without the PU, approximately 80% of OFDM packets have been received correctly, meaning that the FER of about 19.7% is not much lower than FER of 12% for the non-contiguous case without filtering. Therefore, filtering the preamble has not reduced the performance much.

In the second case, when the PU is transmitting, the FER of the secondary user was 20.66%, only slightly higher than the previous FER of 19.7%. Therefore, we can use spectrum-level virtualization successfully to do dynamic spectrum access.

However, as with any dynamic spectrum access scenario, there are additional problems such as sensing the spectrum, or querying a database to know where the PU's signal is. These problems are outside the scope of this thesis, but a large body of work exists on these topics, summarized in [189] and [208].

The use of spectrum-level virtualization for flexible spectrum access can be compared to the nulling of certain subcarriers for OFDM signals such as used in Non-contiguous OFDM (NC-OFDM) scenarios [209]. However, there are key differences to note. Firstly, *any* PHY can be used with spectrum-level virtualization, whereas only OFDM can be used in the NC-OFDM case. Secondly, the PHY layer requires no modifications when virtualization is used, whereas considerable changes are necessary to the OFDM PHY [209]. Lastly, virtualization allows both the nulled spectrum and the PHY technology used to be switched dynamically and flexibly, whereas it is much more difficult to do this in the case of NC-OFDM.

7.9.5 Physical Layer Security

Another application of spectrum-level wireless link virtualization could be physical layer security. Physical layer security refers to the concept of exploiting the physical properties of the wireless channel to encrypt information and achieve secure transmission between nodes [210]. However, we argue that PHY security also includes secure communication through the use of the physical properties of the wireless *signal*.

Security can be achieved in several different ways: computational, physical, and information theoretic [211]. Wireless link virtualization could provide a means of achieving computational security, using the spectrum map as a permutation sequence, also known as scrambling. A similar idea is described in the work [212], however, the benefit of wireless link virtualization is that multiple real or virtual signals can be used, in addition to flexible spectrum use.

The number of possible permutations of the spectrum map depends on the FFT sizes of the input and output streams. It can be calculated using the input stream FFT sizes if the number of inputs is greater than the number of outputs. Otherwise the output stream FFT sizes should be used. We term the vector of FFT sizes $\mathcal{S}$. In general the number of permutations is

$$\text{permutations}(\mathcal{S}) = \frac{\left(\sum_{n=1}^N S_n\right)!}{\prod_{n=1}^N S_n!}$$

where S_n indicates the FFT size of stream n , and N indicates the total number of streams in $\mathcal{S}$. For example, if we have a spectrum map with two inputs of size 64, and one output of size 128, the resulting number of permutations would be

$$\text{permutations}([64, 64]) = \frac{128!}{64! \times 64!} = 2.3951 \times 10^{37}$$

A spectrum map with one input of size 128 and two outputs of size 64 would give exactly the same result.

In addition, a simple modification to the spectrum mapping would allow specific bins to be selected per stream. In other words, rather than systematically mapping each stream from the first FFT bin to the last FFT bin, these bins could also be scrambled. In this case the number of permutations only depends on the overall FFT size and is calculated as

$$\text{permutations}(\mathcal{S}) = \left(\sum_{n=1}^N S_n \right)!$$

where $\mathcal{S}$ can be either the vector of input streams sizes, or output stream sizes.

Demonstration 2: Physical Layer Security

To show how physical layer security can be achieved in practice, we demonstrate the use of wireless link virtualization to achieve security between transmitter and receiver nodes on the testbed. In this demonstration, we use similar flowgraphs to the flowgraphs used in Experiment 2, except that *two* OFDM signals are used, rather than an OFDM signal and a GMSK signal. However, the first OFDM signal uses Quadrature Phase Shift Keying (QPSK) while the second OFDM signal uses Binary Phase Shift Keying (BPSK). We also omit any filtering and resampling operations.

The spectrum map used for this demonstration is shown in Figure 7.25(a). In fact, this is the same spectrum map as used in Simulation 1d in the Chapter 6. This spectrum map has two inputs of size 64 and one output of size 128, and thus there are roughly 2.4×10^{37} possible permutations. The virtual and real spectra for the transmitter are shown in Figure 7.25(b).

As before we measure the correctly received throughput of each signal to determine the frame error rate. The first OFDM signal has an FER of 7.8%, while the second OFDM signal has an FER of 9.91%. The worse performance of the second signal can be explained

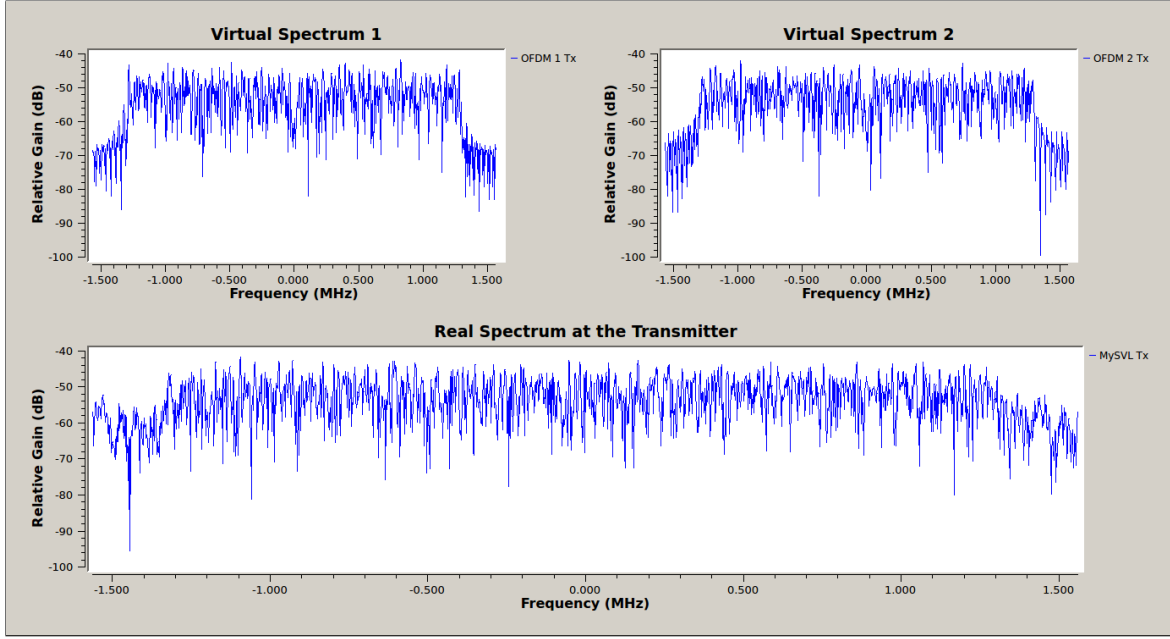
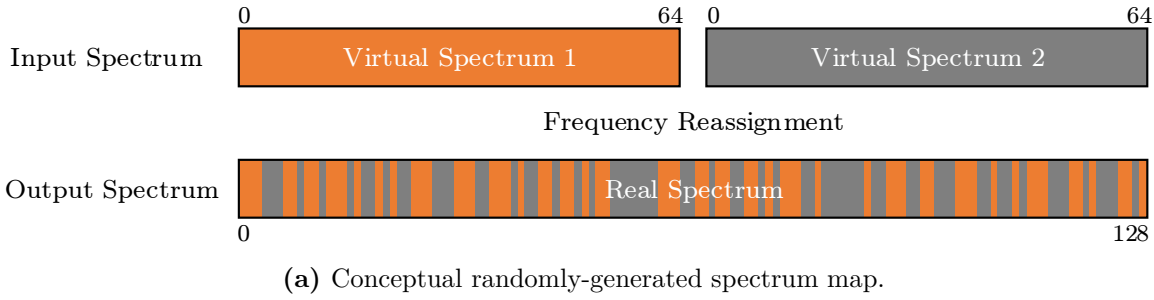


Figure 7.25: MySVL can be used to achieve computational physical layer security, by scrambling the FFT bins. In this example, two OFDM signals are interleaved in a way that only the right spectrum map can descramble the two signals.

by the fact that some frames are dropped at the start of the transmission, and since the first signal has a higher throughput, the effect on total throughput is lessened.

However, although we were able to use wireless link virtualization for physical layer security in the case of two OFDM signals, it is not as easy to scramble the signals when different modulation types are used, as the signals will interfere with each other, rather than being orthogonal to each other.

7.10 Summary

In this chapter we have investigated how isolation at the spectrum-level virtualization can be done in real world conditions, through the use of the CONNECT testbed. We have

shown that contiguous spectrum mapping needs no additional measures, provided the signals do not interfere, and that spectrum virtualization is very feasible. For the non-contiguous case, frame synchronization is required for consistent operation. However, it is better not to use carrier frequency offset estimation and correction as it adds a lot of complexity for very little gain.

We have shown that the correctly received throughput for spectrum-level virtualization is very similar to base operation, and that it is only slightly decreased. Although we did not perform any complexity analysis, we were able to perform complex experiments on commodity hardware, using virtual computing nodes with limited processing power.

Most importantly, we showed some potential applications for spectrum-level virtualization, and we were able to demonstrate these in action. Additionally, it is possible for other researchers to investigate potential 5G applications and interesting research problems, since MySVL is available online and is implemented on the most widely used software radio platform, GNU Radio. One of these problems is the embedding problem which we examine in the next chapter.

8 Spectrum Level Virtualization: The Embedding Problem

8.1 Introduction

As noted in the Theory of Virtualization we developed in Chapter 3, virtualization is a resource mapping consisting of two problems; the first is the isolation problem, and the second the embedding problem, which is the process of allocating virtual resources to real resources. Since the embedding problem depends greatly on the isolation problem, as the method of isolation determines how virtual resources can be used, we first examined the isolation problem in the context of spectrum-level virtualization in the previous two chapters, and implemented methods to isolate virtual spectrum resources.

However, although we were able to isolate virtual resources, we did not yet analyse how we can *map* virtual resources to real resources. This mapping (embedding) is of great significance as it determines how resources are used, and more importantly perhaps, who is allowed to use them, as the embedding problem often consists of multiple competing users. Therefore, in this chapter we examine the embedding problem for wireless link virtualization at the spectrum level.

8.2 The Embedding Problem

The embedding problem is fundamentally a resource allocation problem, which is the distribution of scarce resources to competing users. There are several problems that can occur, described in Section 3.5 in Chapter 3. Figure 8.1 shows the embedding problem visually.

In this thesis, we consider that virtual resources can be offered to one or multiple

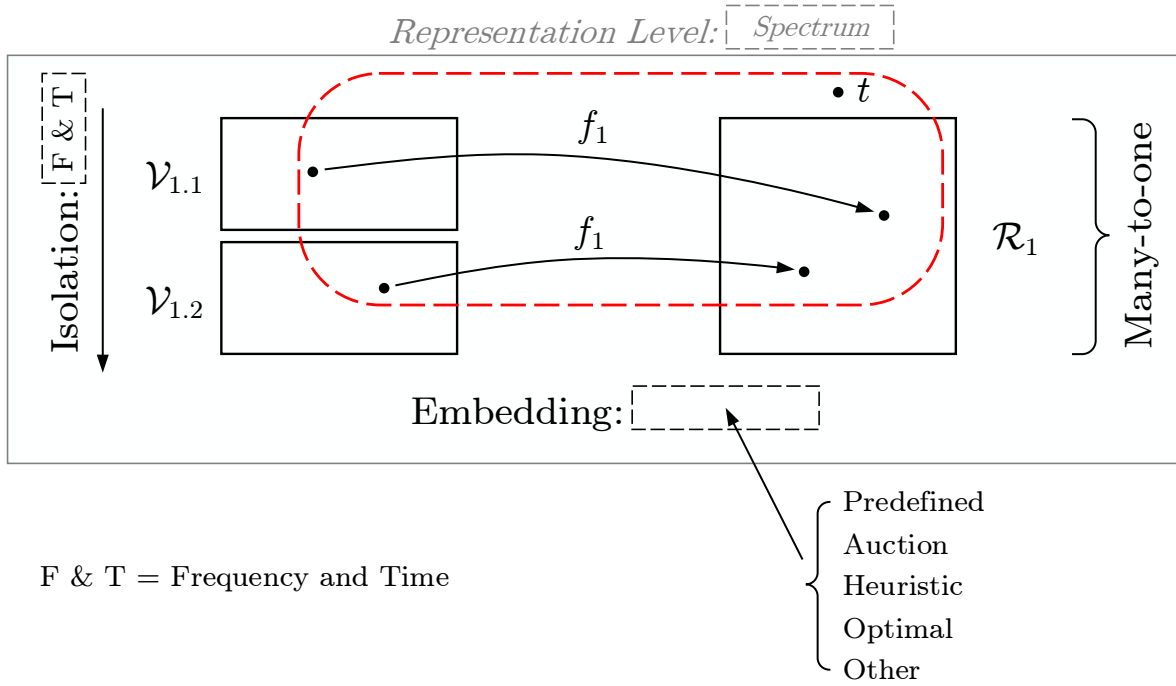


Figure 8.1: In this example, wireless link resources are virtualized at the spectrum level using frequency and time isolation. The embedding problem is about allocating independent virtual resources to real resources, or the specific mapping, f , described previously in Chapter 3, highlighted in red. Many different types of embedding techniques exist, displayed on the bottom right of the figure.

users. The term ‘users’ in this case refers to independent agents, that make decisions on resource use independently. Users can make requests for virtual resources in some manner, and if successful, they receive the use of these virtual resources. Each user has the illusion of full ownership of the resources, meaning that virtual resources can be used for differing purposes.

Depending on the objective(s) that the owner of the real resources wants to achieve, different metrics can be used to determine the optimal resource mapping. Even when there are enough resources to satisfy all of the users’ requests and a potential solution exists, resource allocation can be a complex problem to solve.

In recent times, the embedding problem for wireless link virtualization has received increasing attention in both academia and industry. Several works propose algorithms for embedding, summarized in Table 5.3 in Chapter 5. However, these existing approaches are often inflexible and assign virtual resources in a static manner. In general, works do not address heterogeneous resource use and it is not possible to alter the virtual resources dynamically – in other words, once virtual resources are assigned the assignment cannot be changed.

The downside of static resource assignment is that spectrum resources might not be

used in the most optimum way, and new methods of using spectrum are not possible, which can hinder innovation. Virtualization allows resources to be used in new ways, and dynamic embedding is needed to take full advantage of this. The term dynamic is used to mean embedding algorithms that attempt to re-embed existing virtual resources to achieve more efficient resource allocation and satisfy additional virtual resource requests [6].

Noting the lack of dynamic embedding, the purpose of this chapter is to explore how the virtual embedding process can be made more efficient and dynamic, not only because there is a research gap from an academic point of view and we can make a contribution in this area, but also because dynamicity and flexibility are in keeping with the design principles and the mindset of this thesis.

We begin our exploration of the embedding problem by formulating the optimization conditions for both the static and the dynamic cases. Next, we briefly describe an existing static algorithm and demonstrate how it can be made dynamic. We present a new dynamic greedy algorithm based on selecting the combinations of virtual resources that maximise resource usage.

We compare our proposed greedy dynamic algorithm to the static embedding algorithm and its adapted dynamic version through simulations. In this model, users are able to request virtual resources in real-time from a Infrastructure Provider (InP) based on detailed requirements. The virtual embedding problem is dynamically addressed through embedding algorithms that can re-embed existing virtual resource to achieve more efficient resource allocation and satisfy additional resource requests. We show that our dynamic algorithm improves the real resource usage and significantly reduces the rejection rate of virtual resource requests.

In addition to more efficient resource use, dynamic embedding also enables new applications and functionality. To demonstrate these benefits, we propose a case study which focuses on Public Protection and Disaster Relief (PPDR) services. In this case study, different types of PPDR services, such as voice, video and messaging, are provided by a virtual Public Safety (PS) operator over a commercial Long Term Evolution (LTE) network. Network resources are virtualized, and the spectrum resources are shared between the two operators. We show that through the use of dynamic embedding with prioritized access, PPDR services can be provided reliably during both normal and emergency operation.

Additionally, we develop new embedding algorithms which can lead to new functionality and more efficient resource use, as shown later in this chapter.

8.3 System Model

As noted previously, full isolation is required between virtual resource, so that each user has the illusion of full control over its virtual resources. Wireless link isolation can be achieved by dividing the real resources (also called substrate resources) into orthogonal dimensions to prevent interference [33]. For now, we assume that we are only concerned with a wireless node, where the wireless substrate resources are divided into a number of frequency and time-domain resources, represented as F and T . The total number of real resources available is $F \times T$.

Each virtual link requires a number of frequency and time-domain resources. In this scenario, we assume that virtual resources are contiguous, as it is easier to isolate contiguous resources than non-contiguous resources (see Chapter 7), and moreover in this scenario the full spectrum band is available to the base station, i.e. there is no primary user that must be avoided.

Users can make requests for the virtual spectrum resources, which we term Virtual Resource Requests (VRRs). VRRs can have different priority levels, based on the level of service that a user wishes to achieve. In exchange for providing virtual resources, the infrastructure provider obtains revenue from the users. The revenue depends on the number of resources used, the duration for which they are in use and the priority level, since requests with higher priorities are more likely to be embedded instantly, and therefore should have higher costs.

Resource allocation is performed at regular intervals called timeslots. A number of VRRs can arrive during a timeslot. These requests are stored in a buffer and at the beginning of the next timeslot, the resource allocation is performed for the VRRs in the buffer. The set of VRRs at timeslot ts are represented by $\mathcal{R}_{ts}$. Each request is a set $R = (p, f, t, d)$. The priority level is represented by p , and the frequency and time-domain requirements are denoted by f and t respectively. Each VRR requires an area A of $f \times t$. The duration of timeslots for which the resources are required is represented by d .

A VRR is successful if the requested virtual resources are assigned to real resources. Once a VRR is successfully embedded it will retain the substrate resources until it expires. If a VRR is unsuccessful within a given timeslot, the VRR remains in the buffer and is embedded in the next timeslot if possible. When a VRR has not been successfully embedded for a certain number of timeslots, max_delay , it is removed from the buffer and the request is rejected. The maximum delay can be different for every priority level.

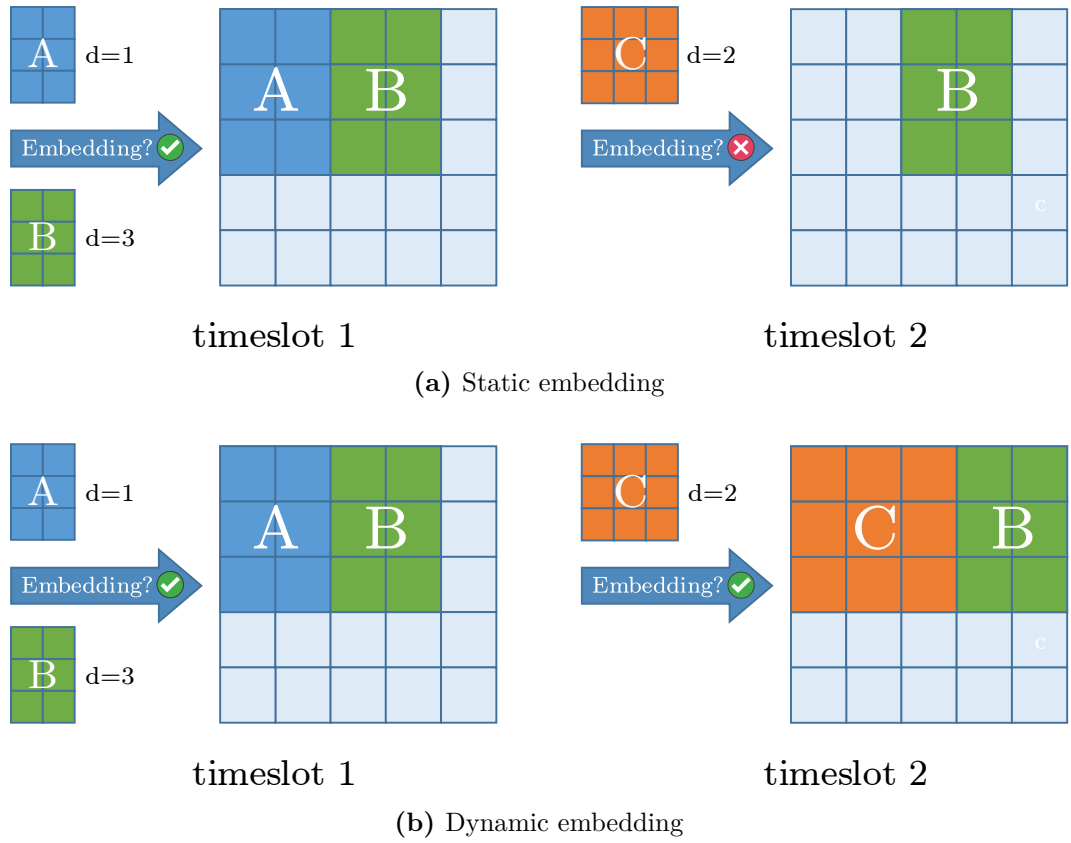


Figure 8.2: Using static embedding, requests A and B can be embedded in timeslot 1, however request C is rejected in timeslot 2. When using dynamic embedding, request C can be accepted in timeslot 2.

8.3.1 Dynamic Embedding

In the static case, successful VRRs will receive the same set of real resources every timeslot for the duration specified. However with this static embedding scheme it is possible that a virtual resource request is rejected due to topology limitations even though there are enough substrate resources available to satisfy it.

The topological limitations of the static embedding scheme are explained in figure 8.2a. Let us assume that the substrate has 5×5 resources available. At timeslot 1 there are two VRRs A and B with area requirements of 2×3 and durations of 1 and 3 respectively. Since there are enough free resources, both virtual resources are embedded. At the next timeslot, virtual resource A has expired and the resources are released. At the same time a VRR arrives with a resource requirement of 3 frequency and 3 time-domain resources. The static embedding method is not able to accept request C because it cannot change the assigned real resources of the embedded request B.

It is possible to overcome this problem by dynamically reassigning the existing virtual resources at every timeslot. The real resources can be used much more efficiently and

additional VRRs can be accepted. Going back to our example, it is clear from Figure 8.2b that using the dynamic embedding approach the substrate resources are used in a more efficient manner.

There are some considerations that must be taken into account when the virtual resource embedding is performed dynamically. Dynamic embedding is more complex than static embedding in the hardware side, and has larger overheads. It is also easier to define the heuristics for static embedding. However it might be beneficial for InPs to use more expensive hardware to support dynamic embedding if the performance is better than static embedding.

8.3.2 Problem Formulation

The allocation of virtual resources to real resources is performed at the start of each timeslot. The new requests with the highest priority are embedded first, followed by the lower priorities. To this end, resource allocation is performed over several stages, corresponding to each priority level. In the static case, we must take into account the resources that are occupied and ensure that VRRs are embedded only in locations that are vacant. In contrast in the dynamic case, we must guarantee that existing virtual resources receive the necessary real resources before embedding new VRRs. By introducing an additional stage for the existing virtual resources, we can ensure that they will be embedded first, followed by the new VRRs in descending priority order.

The embedding of the existing virtual resources and the new VRRs is performed in stages from 1 to the number of priority levels, K . At each stage, k , the embedding for the previous stages must be taken into account. $\mathcal{P}_k$ represents the set of virtual resources embedded in the stages previous to stage k . The set $\mathcal{P}_1$ is always empty as there are no previous stages that need to be accounted for. The set of requests embedded at stage k is represented as $\mathcal{S}_k$. The full embedding for a timeslot is thus found when the embedding for the last stage ($k = K$) has been completed.

The objective at each timeslot ts is to maximize the revenue for the infrastructure provider while guaranteeing that VRRs are embedded in order of priority. We propose a revenue function, $rev(ts)$, which represents the revenue that the infrastructure provider can achieve at every time slot. The revenue collected is based on the number of resources occupied at each timeslot and their priority level. We assume that the physical infrastructure owner can earn more revenue for high priority virtual resources than for resources with lower priorities. The scalar p_s signifies effect of the priority on the revenue earned for request s .

$$rev(ts) = \sum_{s=1}^{|\mathcal{P}_k|+|\mathcal{S}_k|} rev_s(ts) \quad (8.1)$$

$$rev_s(ts) = \begin{cases} p_s Area_s, & \text{if embedded.} \\ 0, & \text{otherwise.} \end{cases} \quad (8.2)$$

The binary variable a_{sij} models the possible embedding locations for each virtual resource request, s . The frequency and time dimensions of the substrate are $i \in \{1, 2, \dots, F\}$ and $j = \{1, 2, \dots, T\}$ respectively. When $a_{sij} = 1$ this means that the virtual resource s is using the real resource block i, j and when $a_{sij} = 0$ it is not using the resource block.

We also add an auxiliary binary variable b_{sij} that is used to signify the starting position of the virtual resource. For any particular VRR s , there can only be one position at which $b_{sij} = 1$.

The dynamic resource allocation problem is by the following optimization problem for every priority level k .

$$\max_{a_{sij}, b_{sij}} \sum_{s=1}^{|\mathcal{P}_k|+|\mathcal{S}_k|} \sum_{i=1}^F \sum_{j=1}^T p_s a_{sij}, \quad (8.3a)$$

$$\text{S. t.} \quad \sum_{i=1}^F \sum_{j=1}^T b_{sij} = 1 \quad \forall s \in \mathcal{P}_k, \quad (8.3b)$$

$$\sum_{i=1}^F \sum_{j=1}^T b_{sij} \leq 1 \quad \forall s \in \mathcal{S}_k, \quad (8.3c)$$

$$\sum_{s=1}^{|\mathcal{P}_k|+|\mathcal{S}_k|} a_{sij} \leq 1 \quad \forall i, j, \quad (8.3d)$$

$$\sum_{i=1}^F \sum_{j=1}^T (a_{sij} - b_{sij} \cdot A(s)) = 0 \quad \forall s \in \mathcal{P}_k, \mathcal{S}_k, \quad (8.3e)$$

$$b_{sij} \cdot A(s) - \sum_{p=i}^{i+f-1} \sum_{q=j}^{j+t-1} a_{spq} \leq 0 \quad \forall s, i, j, \quad (8.3f)$$

$$a_{sij} \in \{0, 1\}, \quad (8.3g)$$

$$b_{sij} \in \{0, 1\}, \quad (8.3h)$$

where $A(s)$ is the area of request s . (8.3b) ensures that the existing requests must be

embedded, while (8.3c) shows that the requests with lower priority can be embedded. (8.3d) provides the constraint that virtual resources must not overlap. (8.3e) ensures that only one b_{sij} can be selected as starting point for each virtual resource and (8.3f) provides the dimensional constraints.

In the static case we must also consider the existing substrate, e_{sij} . This matrix represents the locations of the existing virtual resources which must be re-embedded in the same location. The optimization problem is similar to (8.3) but we also have the additional constraint below to ensure that existing virtual resources are mapped to the same real resources.

$$e_{sij} - a_{sij} \leq 0 \quad \forall s, \forall i, \forall j \quad (8.4)$$

For both the static and the dynamic case, this optimization problem is repeated for all stages 1 through K to find the overall resource allocation.

8.4 Dynamic Embedding Algorithms

The embedding problem described in Section 8.3 is a version of the *multidimensional heterogeneous knapsack* problem [213]. The two-dimensional knapsack problem is a problem in combinatorial optimization and is an NP-hard optimization [214]. In our case we are adding the additional requirement of priority levels, which makes the problem slightly less complex (as there are more constraints in choosing which objects to pack). We also have the orientation constraint, which means that it is not possible to rotate the objects (the virtual resource requests) since they require specific frequency and time dimensions [215]. However, the problem is still NP-hard, and so heuristic algorithms are necessary to achieve feasible solutions.

8.4.1 Static Karnaugh-map Algorithm

The Karnaugh-map based approach in [126] is an example of a heuristic solution. In this work, heterogeneous VRRs are embedded on a two-dimensional substrate by finding Karnaugh-map like regions of vacant real resources. A detailed description is given below.

The aim of the Karnaugh-map algorithm is to maximize the resource occupancy of the substrate and minimize the rejection rate of VRRs. The algorithm attempts to place virtual resources on the substrate in such a manner that the highest number of additional VRRs can be accepted. The best way of achieving this is by clustering the virtual resources together, so that the vacant real resources are contiguous. This allows virtual resources of varying sizes be embedded, maximising the occupied area and minimising the rejection rate.

At each timeslot the set of VRRs that are in the buffer are sorted by priority and by decreasing area. Then the requests are embedded sequentially; the existing virtual resources are re-embedded first and the new requests follow. For each VRR, the Karnaugh map approach is used to find the set of vacant substrate regions with dimensions equal to or greater than the virtual resource's f and t . The smallest region is selected from this list as the location for embedding the VRR. Within this region it is necessary to find the best corner (of the four corners) at which to embed the virtual resource.

The corner that maximizes resource clustering is found using the Embedding Density Index (EDI). The EDI measures the number of borders between free and occupied resource blocks. A high EDI index represents a substrate where the virtual resources are spread out and not clustered together, whereas a substrate with virtual resources that are grouped together will have a low EDI index. The EDI is calculated for each corner, by testing how the new virtual resource affects the clustering of resources. After testing the possible locations, the location with the smallest EDI is chosen as the location to embed the new virtual resource. This ensures that the virtual resource is grouped with the embedded virtual resources as best as possible.

After embedding the new virtual resource requests, the number of timeslots remaining for each of the virtual resources are updated and virtual resources that have expired are removed. Then the algorithm moves on to the next timeslot.

8.4.2 Dynamic Karnaugh-map Algorithm

We adapted the Karnaugh-map algorithm for the dynamic case, by re-embedding existing virtual resources at every timeslot.

8.4.3 Greedy Dynamic Algorithm

We developed a heuristic algorithm known as the Greedy dynamic algorithm. Rather than sorting the VRRs by area, this algorithm considers the embedding of combinations

of multiple VRRs, so that a more efficient embedding can be found. Similar to the dynamic Karnaugh-map algorithm, the input at each timeslot is the set of existing virtual resources and the VRRs in the buffer. The aim is to find the combination of virtual resources that maximizes the number of embedded virtual resources, while guaranteeing that VRRs with high priority are considered first.

As is shown in Algorithm 1, it is first necessary to find the set of possible combinations of VRRs. For each priority level, the set of combinations of higher priority VRRs must be included, but since the higher priority requests are allocated real resources first, the combination that maximises the higher priority requests must be selected (referred to as *previous_combination*). *previous_combination* is known already, and thus it is only necessary to find the set of combinations of VRRs with the current priority level.

Algorithm 1 Dynamic Greedy Embedding Algorithm

Input: $\mathcal{R}_{ts}$ and *previous_combination*

```

1: for all priority levels do
2:   possible combinations,  $P = \{\}$ 
3:    $S =$  subset of  $\mathcal{R}_{ts}$  with current priority
4:   find all combinations,  $C = \binom{|S|}{n}$  for  $n = 1 : |S|$ 
5:   for all  $i = 1 : |C|$  do
6:      $T = \{\text{previous\_combination}\} \cup \{C(i)\}$ 
7:     if area of VRRs in  $T \leq \text{Total\_Area}$  then
8:        $P = [P; T]$ 
9:     end if
10:  end for
11:  sort  $P$  by area of combinations
12:  while no embedding found do
13:    dynamic Karnaugh-map embedding for current combination
      in  $P$ 
14:  end while
15:  previous_combination = current combination in  $P$ 
16: end for

```

Similarly, the combination of existing virtual resources is known at the start of each embedding. This combination can be included at the highest priority level by simply substituting *previous_combination* with the combination of existing virtual resources.

The set of combinations are sorted by decreasing total number of substrate resources used. Although we know that it is possible to embed the combinations based on resource constraints, we cannot be sure that it is possible to embed the combinations based on the dimensional constraints of each VRR. To overcome this, we test the embedding of combinations sequentially until an allowed embedding is found. The algorithm used to

perform the embedding is the dynamic Karnaugh-map; however any other embedding algorithm could be used.

8.5 Performance Evaluation

In this section we measure the performance of the static and dynamic embedding algorithms. The metrics we use to evaluate the algorithms are discussed and we examine how the algorithms compare to the solutions found through Mixed-Integer Programming (MIP) optimization.

8.5.1 Evaluation Metrics

The first metric we use to measure algorithm performance is the revenue the physical infrastructure can expect to achieve from the users.

The second metric that is used is the rejection rate, $r(t)$, which reflects the level of service that the virtual resource users receive from the physical infrastructure provider. The rejection rate per timeslot depends on the number of requests from that timeslot that were not satisfied and were removed from the buffer (after attempting the embedding max_delay times). The resource demands of the request and the duration required is taken into account.

$$r(ts) = \sum_{s=1}^{\mathcal{R}_{ts}} \frac{A_{s, rej} d_{s, rej}}{A_s d_s} \quad (8.5)$$

8.5.2 Simulation Setup

We assign three priority levels; the highest for real-time services such as Voice over Internet Protocol (VoIP), the second for high data rate services such as video streaming and the lowest priority for best effort services such as messaging and downloading.

The arrival of requests is modelled as a Poisson process with average rate λ virtual resources requests arriving per timeslot. The duration of each virtual resource follows an exponential distribution with an average lifespan of μ timeslots. The default values are $\lambda = 3$ and $\mu = 10$. The priority level of each new VRR is distributed uniformly as $U(1, 3)$. The maximum delay, max_delay , is set to 1 timeslot for high priority requests,

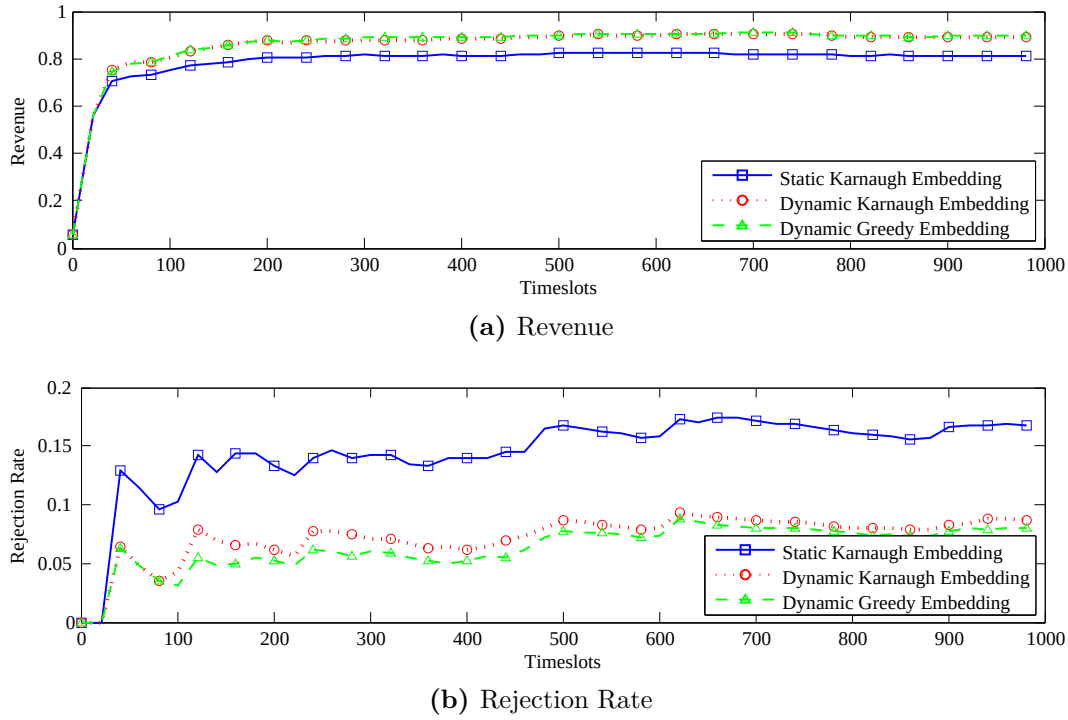


Figure 8.3: The dynamic embedding algorithms provide an increase in revenue for the InP and a large decrease in the rejection rate of virtual resource requests.

2 timeslots for medium priority requests, and 3 timeslots for low priority requests. The dimensions of the substrate are set as $F = 12$ and $T = 12$.

The area of each request is initially modelled as a uniform distribution with $U(1, 3)$ for the frequency dimension and $U(1, 3)$ for the time dimension. We run the simulation for 1000 timeslots. The function p_s which represents the relative cost for different priority levels is modelled as follows $p_s = \alpha$ for the highest priority, β for the medium priority and γ for the lowest priority. We set $\alpha = 0.5$, $\beta = 0.3$ and $\gamma = 0.2$. The total number of resources requested over the course of the simulation is 141,683 while the total available resources is 144,000.

8.5.3 Results and Discussion

Dynamic embedding achieves better performance than static embedding

In Figure 8.3 we compare the dynamic embedding algorithms and the static algorithm. Fig. 8.3a shows the average revenue while Fig 8.3b shows the average rejection rate of VRRs. After some initial fluctuation the graphs smooth off to the long-term values. It is clear that the dynamic embedding algorithms outperform the static algorithms in terms of revenue and rejection rate.

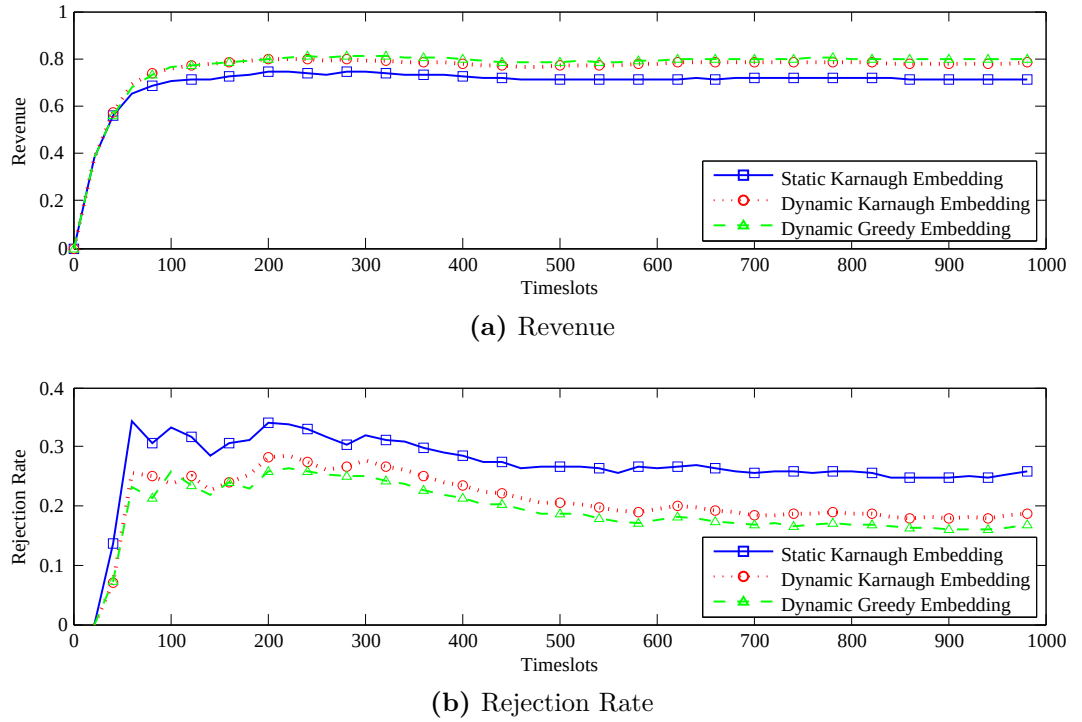


Figure 8.4: A larger average request size leads to smaller revenues, and a higher number of rejected requests.

It is interesting to note that the rejection rate for the dynamic algorithms is roughly 50% lower on average than the static case but that the revenue is only 8% better in the dynamic case. We can also note that the dynamic greedy algorithm achieves a slight increase in performance compared to the dynamic embedding. This shows that the use of dynamic algorithms benefits the users of the virtual resources as the rejection rate decreases significantly, but that for the infrastructure owner the additional revenue obtained is quite small.

Accuracy of resource request prediction

Until now we have assumed that the users making the VRRs know exactly what resources are required for each VRR. This may not be the case in practice. The accuracy of the predictions made by the users influences the performance of the embedding algorithms. If the users can make very accurate predictions then it is possible to have many requests with small resources requirements. However if it is not possible to predict accurately, it may be necessary to over-provision resources and make large requests.

We examine the case when the average resource requirements of the VRRs are higher, but the total number of resources requested remains similar. We set the resource requirement for each dimension to $U(2, 4)$ and reduce λ to 1.5. The total number of resources

requested is 141,655, which is very similar to the default scenario in which requests are smaller but more numerous.

From Figure 8.4 we observe that the rejection rate is 1.7 times higher on average than the default scenario. The revenue is 13% lower on average. Fig 8.5 shows the mean rejection rate and the standard error for the various priority levels and scenarios. From this figure, we can observe that a large number of high priority requests are rejected in the scenario where the requests are large. Since the high priority requests are the most valuable, this has the most impact on the revenue. We can conclude that when the users are not able to make accurate predictions and the resource requests are large, this has a negative impact on both the rejection rate and the revenue.

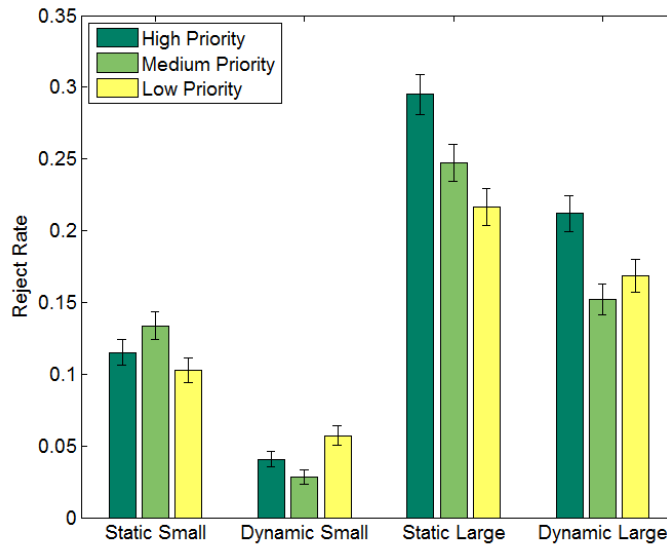


Figure 8.5: Detailed rejection rates for the small and large request sizes show that the high priority rejection rate is much higher for large request sizes, relative to the other rejection rates.

Comparison to optimal solutions

We find the optimal solutions for both the static and dynamic cases by applying MIP optimization to the problem formulation we developed in section 8.3. However, the optimal solutions are computationally intensive to find, which is the reason for using heuristic algorithms such as the greedy embedding algorithm. Therefore we can only find the dynamic optimal solution for a 7×7 substrate, as larger substrates are too computationally complex to use.

The static optimal solution is even more complex than the dynamic optimal, and can only be found for small substrates (roughly up to 5×5). Because small substrates

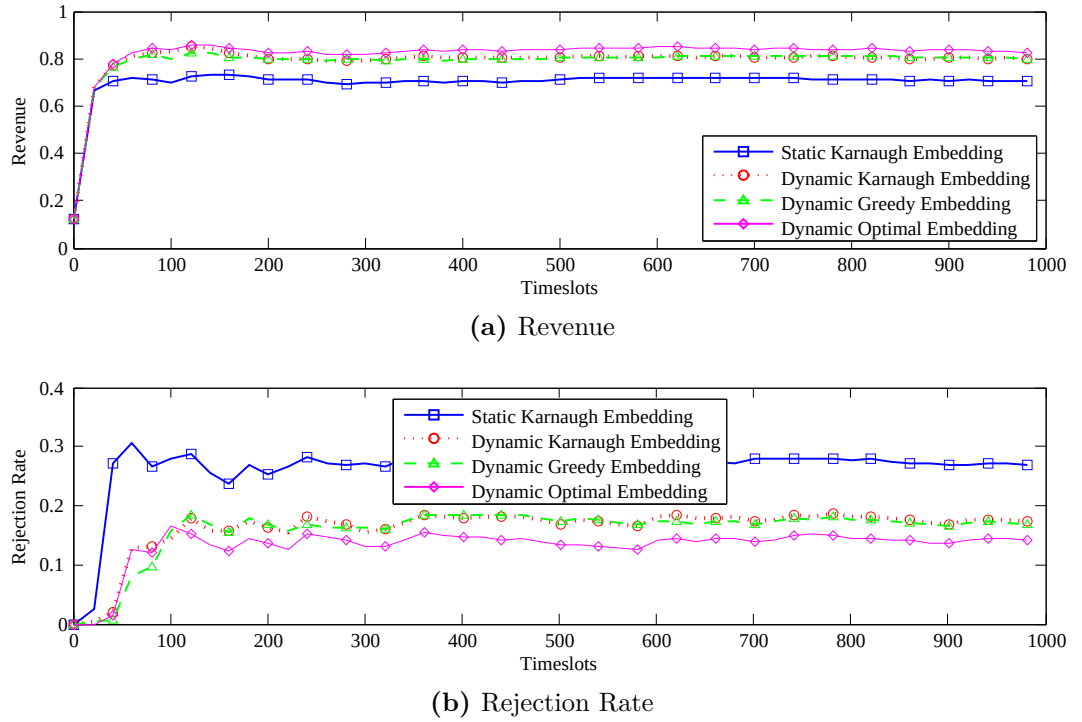


Figure 8.6: Comparison of embedding algorithms to the optimal embedding, showing that although the optimal has a better performance by about 2-3% than the dynamic greedy algorithm, our algorithms perform reasonably well in comparison.

can vary greatly in terms of revenue and rejection rate based on the particular request dimensions, we do not show the static optimal solution for a small substrate.

Figure 8.6 compares the algorithms to the dynamic optimal solution for a 7×7 substrate, where in total 47,568 resources are requested. As we can see, the optimal embedding is about 2-3% better than the greedy algorithm both in terms of revenue and rejection rate. However, the optimal embedding solution is very complex to compute, especially as the size of the substrate increases.

Priority costs

The total revenue that can be achieved is influenced by the relative cost assigned to each priority level. However, the cost of each priority level also affects the number of VRRs that are requested at that priority level. In the absence of a market for wireless link virtualization resources the optimum relative cost of each priority level cannot be known. In our model, since the demand for each priority level is uniform, the values assigned to p_s do not influence the total revenue. Therefore, without the addition of a model to simulate the market for wireless link virtualization resources at different prices, it is more useful to measure the number of resources occupied, rather than the revenue.

Complexity of the embedding problem

As mentioned briefly, the multidimensional heterogeneous knapsack problem is NP-hard. In our case we investigated a two-dimensional where our dimensions were frequency and time, and we showed that the heuristic algorithms developed were able to arrive at a suboptimal solution within a reasonable compute time (on the order of a few seconds per timeslot). However, additional dimensions exist for spectrum, such as the power used, the polarization of energy, or the code used in a Code Division Multiple Access (CDMA) system. Since the three-dimensional heterogeneous knapsack problem is already much more complex than the two-dimensional problem (the multi-dimensional knapsack problem is strongly NP-hard [215]), we do not consider the possibility of using more than two dimensions simultaneously. For more information on the complexity of the multidimensional knapsack problems see [216].

8.5.4 Conclusions

We examined the performance of the static and dynamic algorithms in an idealized context, and compared the algorithms in terms of rejection rate and revenue. However, embedding requires a context, since resource allocation is concerned with different conflicting choices, and it is not meaningful to use an ideal scenario where the choices made are of no consequence. Without competing users, there was little use in measuring the revenue, which was simply proportional to the resource efficiency. Therefore, it is necessary to apply virtualization to a more meaningful scenario to investigate the embedding problem properly.

8.6 Application of Dynamic Embedding: Public Safety

In the previous sections, we have examined the performance of static and dynamic embedding in an idealized scenario. However, it is not meaningful to perform virtual resource embedding without a context, as embedding is a form of resource allocation, and as such involves compromises between different objectives. To this end, in the remainder of this chapter, we examine virtual resource embedding in an applied context, namely the context of PPDR services.

8.6.1 Introduction to PPDR networks

Mobile broadband communications have great potential to improve the efficiency of Public Protection and Disaster Relief operations by enabling applications such as real-time access to high-resolution maps and floor plans, on-field live video transmission from helmet cameras to a central unit, telemedicine, etc. [217] [218]. Driven by this demand, significant changes are taking place in the professional mobile radio industry. While existing Professional/Private Mobile Radio (PMR) technologies, such as TETRA, TETRA-POL, DMR, APCO P25, GSM-R and others, have been very successful in delivering business and mission critical voice and narrowband data services in these professional sectors, these technologies are not well suited to support higher data rate applications.

In this context, the adoption of mainstream commercial technologies such as LTE/LTE-Advanced for PPDR mission critical communications is gaining strong momentum [219]. Indeed, many national and international PPDR organizations have endorsed or are considering LTE as the next generation technology either to augment their existing systems, or to provide a future migration path. The economies of scale of LTE are expected to bring down the costs of private mobile radio equipment. Establishing common technical standards for the PMR and commercial domains offers significant opportunities for exploiting the synergies between these two domains through network and spectrum sharing principles [220].

LTE technology was originally conceived and designed as a flexible and spectrally efficient mobile broadband technology for the commercial domain. However, work is in progress within the 3GPP to extend the use of LTE to critical communication applications and to deliver future services to PPDR users [221].

In most parts of the world, the prevailing model that has been used for the delivery of mission critical narrowband PPDR communications (voice-centric and low data rate services) is based on the use of dedicated networks; networks deployed for PPDR use that are mainly privately built and operated with the specific purpose of serving the communications needs of a limited number of agencies. This delivery model, while undoubtedly offering full control and high availability of communications resources to PPDR users, has resulted in a niche market with far less innovation and higher prices for communication equipment in comparison to the commercial domain.

Given that PPDR communications systems are fundamentally funded from constrained public authorities' budgets (i.e., taxpayer's money), PPDR communications should strive to achieve the most cost-effective solution. Without compromising the high control, security and resilience standards required by PPDR communications, future broadband

PPDR data services must be delivered through optimal deployment mechanisms, in a cost-effective manner with a sustainable business model [221].

Acknowledging that while the most suitable model may differ across countries and regions because of differing existing conditions and interests, we consider the delivery of mobile broadband services to PPDR users over public-access LTE-based networks. These networks are run by commercial operators and the PPDR users share the infrastructure with the rest of commercial traffic.

Despite some reluctance to use commercial networks from the PPDR sector since it is believed that current commercial networks cannot satisfy many of their requirements (see [222], [223] for example), there is increasing consensus that these infrastructures undoubtedly have a role to play in critical communications solutions. As commercial broadband networks become an important part of society's infrastructure, they can enable PPDR users to experience the benefits of broadband communications [224], [225].

In terms of materialization of these principles, [220] presents a system architecture that enables PPDR service access across a number of LTE-based dedicated and commercial networks. The system is implemented through a Mobile Virtual Network Operator (MVNO) model such that no dedicated LTE networks are deployed for PPDR. As an example of practical realization, ASTRID, the national operator of radio communications, paging and the dispatching network for emergency and security services in Belgium, launched a broadband data service called Blue Light Mobile [226] in Spring 2014. This service allows PPDR users to use the commercial 3G networks for data-centric applications. To that end, ASTRID takes on the role of an MVNO that offers services via the mobile networks of the three national operators in Belgium.

Given this background, we investigate whether a virtual PS operator can exist on a virtualized commercial LTE network using our previously developed embedding techniques. We compare several methods of allocating spectrum resources between virtual operators at peak times and examine how this influences differing traffic services. We examine the reliability of such a network for PPDR services, and the benefits to the commercial network owner.

8.6.2 System Overview

The proposed scenario is very similar to the scenario we previously examined, except that we specifically consider LTE. We consider the virtualization of a single evolved Node B (eNodeB) in an LTE network. In this model, a hypervisor is added on top of the physical hardware of the eNodeB that manages the resources for several virtual eNodeBs

[100]. The hypervisor is also responsible for assigning virtual LTE spectrum resources (known as Physical Resource Blocks (PRBs) ⁸) amongst the virtual links. Since the PRBs consist of two orthogonal dimensions, frequency and time, full isolation can be ensured between virtual links.

Several Virtual Network Operators (VNOs) can exist on a real eNodeB, each with their own virtual eNodeBs. For now, we assume that sufficient computational resources exist to satisfy the needs of the virtual eNodeBs, and that the computational resources are fully isolated. We are interested only in the virtual PRB assignment.

At regular time intervals or timeslots, the hypervisor performs the PRB assignment. We assume that the time between consecutive timeslots is such that each of the PRBs have the same average throughput and are equally adequate in the long run. This means the hypervisor can assign any of the PRBs to a virtual link rather than assigning specific resource blocks.

At any time, VNOs can make VRRs to the hypervisor on behalf of their users for r PRBs. To ensure that this model applies to both uplink and downlink resource allocation in LTE, and to allow for easier isolation, the virtual PRBs resources allocated to each user must be contiguous, [227]. Therefore, upon receiving a Virtual Resource Request for r resources, the hypervisor maps this to a rectangle $f \times t$, with time and frequency dimensions f and t , respectively, which ensures that the virtual Physical Resource Blocks assigned to each user are contiguous.

In addition to specifying the number of resources desired, VRRs also specify a duration, d , for the number of timeslots that the resources are desired. A priority level, p , is also assigned to VRRs, which allows virtual network operators to prioritise requests depending on the service type, s . Thus each request for a service s , $VRR(s)$, has three parameters: $\{p, r, d\}$.

Arriving VRRs are stored in a buffer until the next embedding timeslot. At each timeslot, an attempt is made to allocate resources to all Virtual Resource Requests in the buffer, within the constraints of the total resources available, $F \times T$, where F and T are the frequency and time dimensions of the substrate. If there are enough free resources to embed a VRR, the request is embedded and deemed successful. The virtual operator that made the request can then use the resources to satisfy a user service for the specified number of timeslot (d). If a VRR is not successfully embedded during a timeslot, it remains in the buffer and is embedded in the next timeslot if possible. A VRR that is not embedded for a number of timeslot, max_delay , is removed from the

⁸The use of the word ‘physical’ here is unfortunate, as the resources are actually in the abstract domain, see Chapter 2.

buffer and the VRR is rejected. max_delay can vary for different services and priority levels.

Priorities during Normal and Emergency operation

The hypervisor has two modes of operation: normal operation and emergency operation. During normal operation a virtual resource request from a given service VRR(s) has an associated priority level p . When an emergency occurs, the priority levels can be changed on a per-service and per VNO basis. Thus, for example, during normal operation the Public Safety VNO may be treated in the same manner as any other commercial operator, but during emergencies, certain services of the Public Safety VNO can be given higher priority.

8.6.3 Investigated Scenario

The scenario we are investigating is that of a commercial LTE-based network, where the Infrastructure Provider has virtualized the mobile resources and Virtual Network Operators can request these resources as needed. In our case there is one PS virtual network operator and one commercial VNO that share the physical network.

Evaluation Metrics

The metrics of interest in this scenario are the percentage of Virtual Resource Requests that are rejected per service and the percentage of total resources used per service at every timeslot. The rejection rate for a timeslot is calculated as the sum of $r \times d$ of rejected VRRs, divided by the sum of $r \times d$ of all VRRs. The first metric shows the importance of reliability from the perspective of the VNOs, while the latter metric provides the InP with a method of measuring how efficiently its resources are used, and thus how much revenue it can obtain. Based on these metrics the InP could decide whether additional investment in physical infrastructure is required, and virtual operators can discover whether the InP provides a satisfactory service level.

Traffic Modelling

We model three common types of services that each operator provides to its users, namely voice, video, and messaging traffic, corresponding to 3GPP's Quality of Service Class Identifiers (QCI) 2, 3, and 6 respectively [228, Table 6.1.7]. The number of resources,

r , required for a service is modelled as a uniform distribution $U(min, max)$, and this is then mapped to a permutation of the smallest rectangular area covering at least r PRBs. The duration of each VRR, d , follows an exponential distribution with an average lifespan of μ timeslots.

We model the three services as follows:

$$\begin{aligned} \text{VRR}(\text{voice}) &= \{p, d = 30, r \sim U(min = 1, max = 2)\} \\ \text{VRR}(\text{video}) &= \{p, d = 10, r \sim U(min = 8, max = 25)\} \\ \text{VRR}(\text{msg}) &= \{p, d = 3, r \sim U(min = 1, max = 8)\} \end{aligned}$$

The maximum delay, max_delay , is set to 1 timeslot for voice service requests, 2 timeslots for video services, and 4 timeslots for messaging services. The values for these parameters reflect realistic conditions to the best of our knowledge.

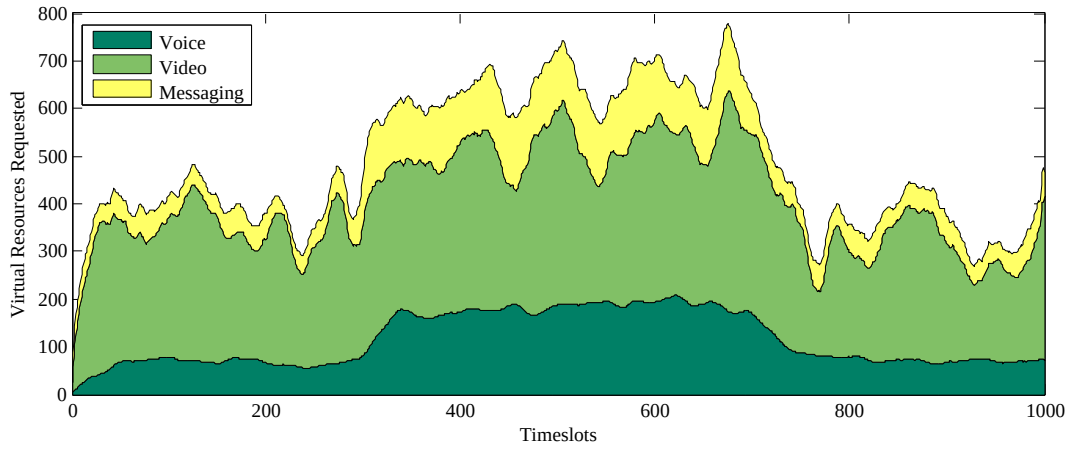
Service Priorities

During normal operation, the VRRs made by the PS and commercial VNOs are considered as having equal importance and priority levels are based only on the service type: the highest priority is for voice requests, next is video and then messaging. When an emergency occurs, all PS traffic is given higher priority than commercial traffic, which retains the same priority as before. Regarding commercial traffic, messaging traffic is considered more important during an emergency than video traffic. To this end, higher priority is given to voice, then messaging and eventually video.

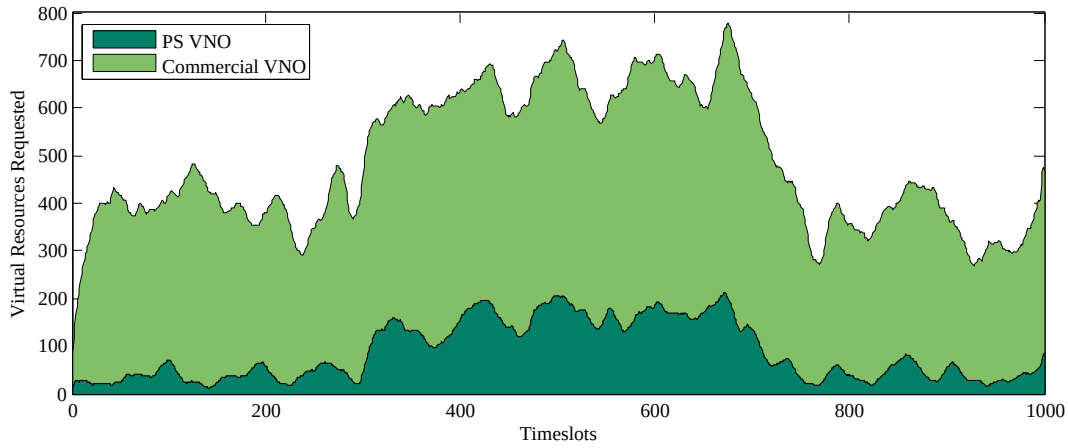
Simulation setup

In this scenario the dimensions of the substrate are set to $F = 20, T = 20$. The resource embedding is performed for 1000 timeslots. Initially, normal, day-to-day operation is assumed for the hypervisor, however after 300 timeslots an emergency occurs. This means that the priority levels of VRRs change as described above. This emergency situation lasts for 400 timeslots, after which the hypervisor switches back into normal operation.

The number of VRRs of each service arriving per timeslot is modelled as a Poisson process with the following average rates λ requests per timeslot: For normal operation, it takes the values 1.4, 1.4, and 3 for commercial voice, video and messaging respectively, and the values 0.14, 0.14, and 0.3 for PS voice, video and messaging. These values offer



(a) Resources requested per timeslot by traffic type.



(b) Resources requested per timeslot by operator.

Figure 8.7: Virtual resources requested per timeslot by traffic type (a) and by operator (b). During an emergency (timeslots 300-700), the total number of resources increases by a factor of 5 for the PS VNO, while increasing by a factor of 2.5 for the commercial VNO.

a realistic approach, since the total percentage of each service type is similar to what is found in the literature and in real systems today [229]. Regarding the traffic share between commercial and PS, it assumes that PS represents 10% of the traffic.

When an emergency occurs, the number of VRRs increases significantly. The PS now requires many more resources and we multiply the values by 5 for all PS services. The commercial voice and messaging traffic are also assumed to increase by a factor of 2.5, while we consider that the video traffic remains constant. The number of virtual resources requested per timeslot is shown in Figure 8.7, both in terms of the traffic types (a) and in terms of the virtual operators (b).

The performance achieved in terms of rejection rate per timeslot is depicted in Figure 8.8. Regarding the performance of the embedding algorithms, it can be observed in (a) that the dynamic embedding achieves 0% rejection for all the offered PS traffic during

the emergency, while the static algorithm rejects about 10% of the PS VRRs. Since lower priority has been assigned to the commercial traffic during the emergency, higher rejection rates (in the order of 30% - globally across all commercial services) are observed. Based on these rejection rates, PS communication can be achieved reliably over this system.

In more detail, regarding the treatment of the different service flows during the emergency, (b) reflects that all voice traffic from both PS and commercial is served. For video service, (c) reveals that the lowest priority given to commercial video traffic during the emergency leads to high rejection rates, in the order of 70%. Regarding PS video traffic, it is observed that the dynamic embedding algorithm is able to provide almost 0% rejection rate, while the static embedding leads to substantial rejections (above 20%). Finally, for the messaging service, (d) shows that rejections only occur for commercial traffic and the dynamic embedding algorithm. This is because the dynamic embedding has led to increased allocation of resources to PS traffic.

8.6.4 Performance Results

We compare the performance of the embedding algorithms in normal and emergency operation and examine whether virtualized LTE networks can offer adequate service for PS.

The performance achieved in terms of resources used per timeslot is depicted in Figure 8.9. The comparison of (a) and (b) illustrates how the dynamic algorithm is able to exploit almost 100% of the resources during the emergency, while the static embedding exhibits lower resource exploitation. During normal operation, the resource usage does not reach 100% because the offered traffic during this period is below the maximum network's capacity. It can also be observed that the capacity share for the PS during the emergency increases (it represents around 40% of the traffic served during the emergency) compared to the normal operation. This is because of the PS traffic increase during the emergency period (i.e., higher requested load) as well as the higher priority assigned to PS traffic (i.e., higher served load). The distribution of the network's capacity across services is shown in (c) and (d). Both the static and dynamic embedding algorithms devote almost 50% of the capacity to voice during the emergency. However, the dynamic embedding is able to allocate a substantially larger share of video traffic (about 25% of the total traffic) compared to the static approach, at the expense of a slight reduction of messaging traffic. In any case, as shown in (d), messaging traffic for the dynamic embedding benefits from the ability of the algorithm to reach almost 100% resource occupation during the emergency period.

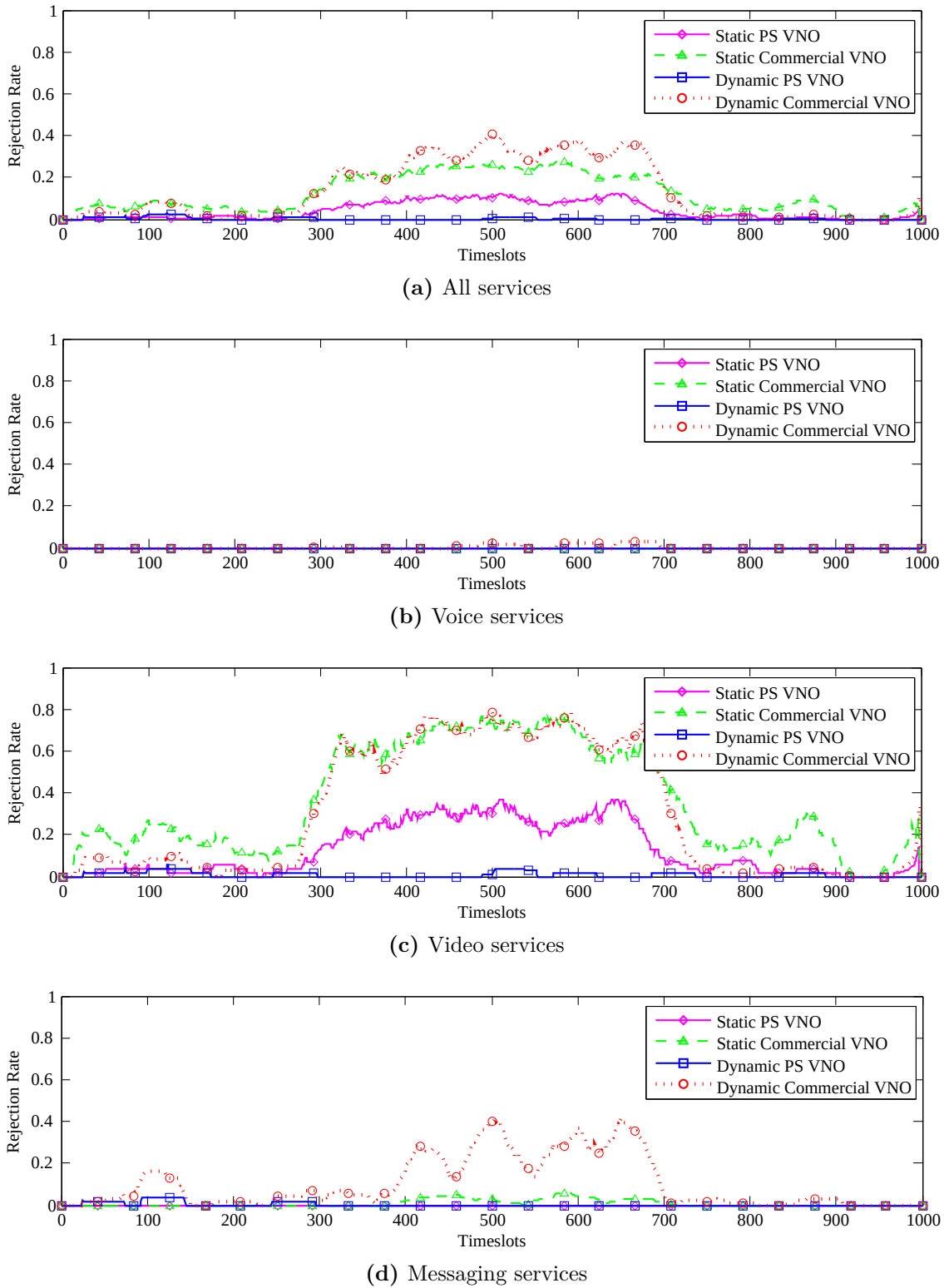
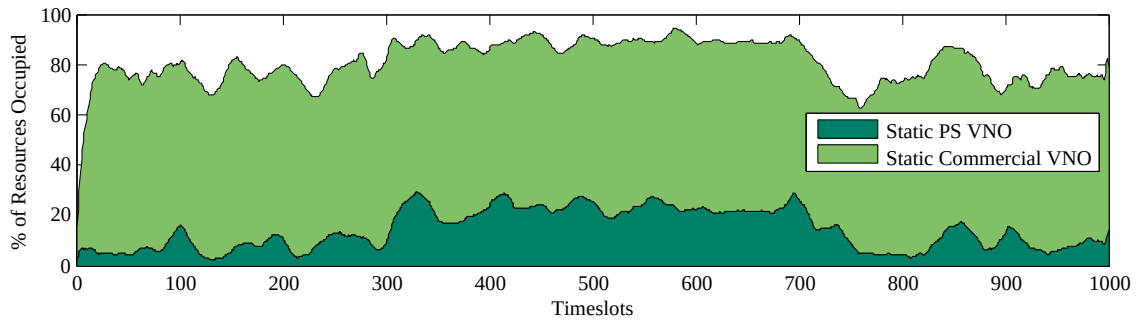
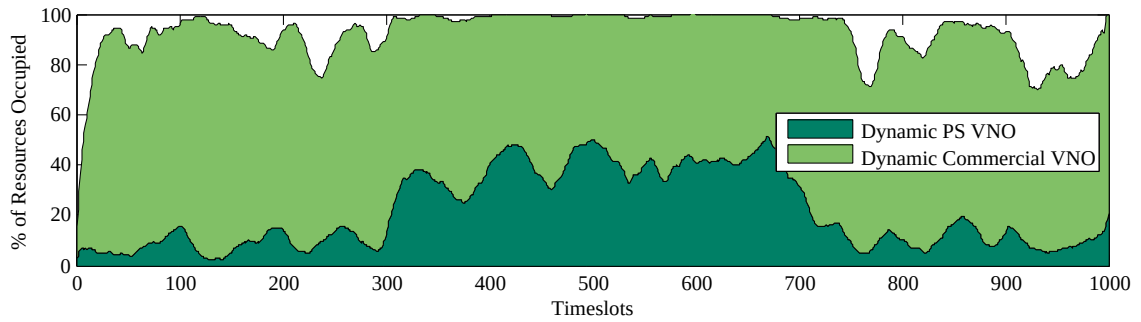


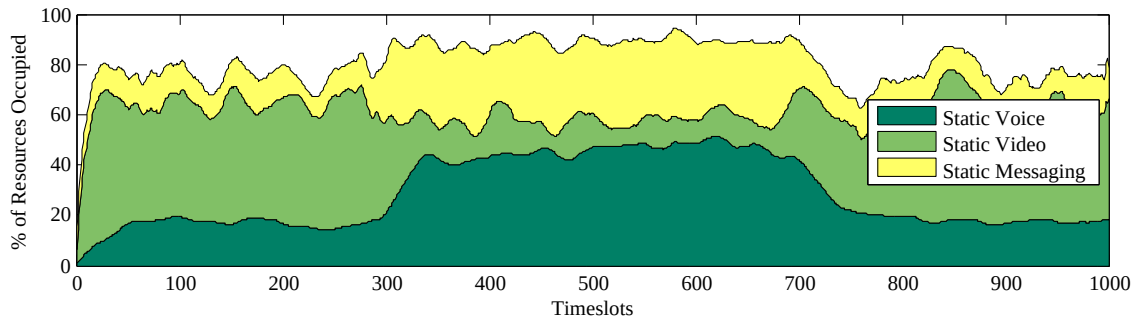
Figure 8.8: Rejection rates per traffic type and operator. Even during an emergency (timeslots 300-700), negligible rejection rates for the Public Safety services can be achieved for all services (a) using the dynamic embedding algorithm, while for the commercial operator voice services (b) are completely reliable, messaging services (d) are quite reliable, but video services (c) are unreliable.



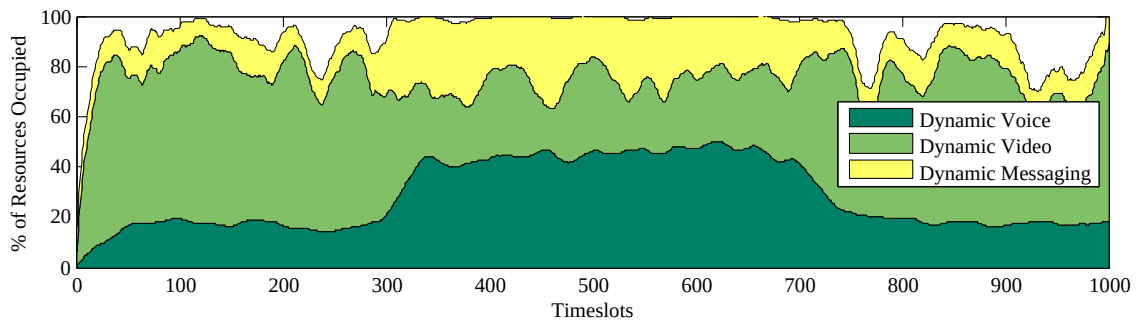
(a) Static embedding VNO split



(b) Dynamic embedding VNO split



(c) Static embedding traffic service split



(d) Dynamic embedding traffic service split

Figure 8.9: Compared to the static embedding (a), the Public Safety operator receives a significantly higher share of total resources when dynamic embedding is used (b). This is mostly due to the low number of video services that are embedding in the static case (c), in comparison to the dynamic case (d).

8.6.5 Conclusions

In this section, we proposed an applied context for virtual wireless link embedding at the spectrum level, namely the scenario where a virtual public safety operator shares a commercial LTE network with virtual commercial operators. We analysed the rejection rate and the percentage of resources occupied for the static and dynamic embedding algorithms. We examined the reliability of such a system in normal and emergency situations for differing traffic services, and showed that it is possible to provide the public safety operator with reliable communication during emergencies, since almost zero services were rejected. The impact on commercial services was also investigated, and though the performance dropped during emergencies, the most crucial services (voice and messaging) were very reliable.

These promising results motivate further studies, such as the formulation of utility functions to capture users' acceptance of the provided service levels during normal and emergency conditions. This would allow more complete design and parametrization of the embedding algorithm. Additionally, the proposed public safety virtual operator model can be detailed further according to actual/future LTE architectures and 3GPP-based prioritization and Quality-of-Service control mechanisms.

8.7 Summary

In this chapter, we examined the embedding problem for wireless link virtualization. Although there are many aspects to the embedding problem, we focused on the area of dynamicity, and the benefits of dynamic embedding over static embedding. We first formulated the static and dynamic optimization problems for a two-dimensional substrate including multiple priority levels. Next, we proposed a dynamic embedding algorithm and modified an existing static algorithm to make it dynamic. We compared these algorithms to the optimal in terms of rejection rate and revenue in an idealized scenario.

Recognising that embedding without a context is not meaningful, we showed how virtualization can be applied to a meaningful scenario, where PPDR services can coexist with commercial services over an LTE network. Examining the embedding problem in this context enabled more meaningful discussion and reinforced the benefits of dynamic embedding.

9 Conclusions

9.1 Introduction

In this final chapter, we summarise the main findings of the thesis and discuss how the approach we have taken in this thesis has enabled us to arrive at these findings. Finally, we describe some of the potential research directions which stem from the systematic approach to virtualization.

9.2 Summary of the Thesis

We started the thesis with a plethora of examples to demonstrate how the term virtualization is used inappropriately, both in common usage and in technical literature, and showed how the imprecise use of the term creates much confusion. One of the main goals of this thesis has been to answer the first research question: to clear up the confusion surrounding virtualization. We believe that clear, precise definitions are needed for 1) finding relevant research problems to study, 2) avoiding repetition and repeating work already done, and 3) understanding new applications of research and new possibilities.

To arrive at a clear definition of virtualization, in **Chapter 2 Prerequisites to Virtualization** we first described Abstraction/Representation (AR) theory, which explains how the physical and abstract domains are connected. Building on the existing theory, we extended it to include multiple levels of representation. We found that AR theory provides a elegant method of defining hardware and software while also enabling us to realise how representation applies to virtualization.

In the next chapter, **Chapter 3 A Theory of Virtualization**, we used AR theory to propose a general theory of virtualization. We provided a clear, structured analysis of the various aspects of virtualization, and classified virtualization into four types. Additionally, we described the often overlooked concept of recursion for virtualization,

and argued that it is one of the defining properties of virtualization. We summarized the theory in a definition of virtualization, which we believe explains virtualization in a clear and consistent manner. We reinforced the validity of the theory by showing how it can be applied to an existing virtualization technology.

Having gained a more clear understanding of virtualization, the second goal of the thesis was to use the definition to examine current problems in wireless network virtualization. In **Chapter 4 Virtualization in Wireless Networks**, we described the development of virtualization technologies, and showed how these technologies are relevant for wireless network virtualization. The theory of virtualization enabled to look at these technologies in a new manner, and we discovered several insights which have not been mentioned elsewhere, to the best of our knowledge. One such insight is the distinction between Radio Frequency (RF) front-end virtualization, wireless link virtualization, and wireless network virtualization, which are often interchanged in existing literature.

Speaking of the existing literature, in **Chapter 5 Survey of Wireless Link Virtualization** we used the precise definition to aid us in examining the existing literature on wireless network virtualization. Foremost, we developed a test for virtualization which allows us to distinguish between techniques which are virtualization, and techniques which are resource allocation but not virtualization. There are many new insights which we discovered throughout this chapter as a result of the rigorous definition of virtualization, for example, we discovered that most work on wireless link virtualization to date has focussed on the information level of representation, and there is a significant lack of implementations at the spectrum level.

As a result of the insights gained in Chapter 5, we decided to focus on wireless link virtualization at the spectrum level, as we believe there are many advantages that can be gained from virtualization at the spectrum level.

Chapter 6 Spectrum Level Virtualization: The Isolation Problem Part 1 investigated the isolation problem, bearing certain design principles in mind, influenced by the theory of virtualization. We first looked at existing implementations to see if we could reuse these to analyse isolation methods. Concluding that this was not possible, we decided to develop our own implementation, the Many-to-many Spectrum Virtualization Layer MySVL, and we designed MySVL to be as general and flexible as possible. We ran simulations to show that the implementation was working as desired, and demonstrated the features of MySVL.

Since the isolation problem is fundamentally about maintaining independence between virtual resources, it can only truly be analysed through real-world testing. Therefore, in **Chapter 7 Spectrum Level Virtualization: The Isolation Problem Part 2**

we executed experiments on the CONNECT testbed. We investigated the measures needed to perform isolation over-the-air, and illustrated the size of necessary guard bands between virtual resources. We were able to successfully achieve wireless link virtualization at the spectrum level for the contiguous case. Next, we examined the non-contiguous case, concluding that frame synchronization was necessary. Having added a preamble and developing corresponding measures at the receiver, we were also able to perform non-contiguous link virtualization. Then, we proposed a method to correct the carrier frequency offset, but although we were able to do this, we realised that the achieved gains were not worthwhile. Finally, we proposed a range of potential applications for wireless link virtualization at the spectrum level, and demonstrated several applications such as physical layer security.

Having successfully performed isolation at the spectrum level, in **Chapter 8 Spectrum Level Virtualization: The Embedding Problem** we analysed the embedding problem. We observed that almost all existing embedding techniques have been static, and therefore, and in keeping with the design principles of the thesis, we proposed the dynamic embedding problem. We formulated the optimization problem for both the static and the dynamic case, and proposed several dynamic algorithms. We showed that dynamic embedding achieves better efficiency and service to users than the static case. Next, since embedding is meaningless without a context, we applied virtualization to the potential scenario of a virtual public safety operator sharing a commercial Long Term Evolution (LTE) network to show the benefits of both virtualization and dynamic embedding.

9.3 Lessons Learned

In this section we discuss the overall lessons learned and debate the merits of the research approach taken in this thesis.

The starting point of this thesis was the confusion and lack of clarity surrounding virtualization, and before performing research on Wireless Network Virtualization (WNV) or Wireless Link Virtualization (WLV), we first wanted to understand virtualization thoroughly. There are a number of benefits of this approach, described below.

9.3.1 Structured Analysis

By developing a clear understanding of virtualization, we were first able to categorise existing computer and network virtualization techniques in a structured manner, using

consistent language. While doing this, we recognised that wireless link virtualization is a prerequisite to wireless network virtualization, and also realised the distinction between WLW and RF front-end virtualization.

Additionally, a clear understanding of virtualization allowed for a structured analysis of the research on wireless link virtualization that has been done to date, and enabled us to see what is missing in current work. This is of significance, as overlooked problems can provide new opportunities to provide new services and increase efficiency. As an example, almost all current work on virtualization is focussed on splitting resources rather than aggregating resources, and this constricted vision of virtualization results in missing out on new techniques, applications and services.

9.3.2 Use of Multiple Representation Levels

We found that only approximately 41% of the works that self-identify as virtualization can actually be considered virtualization according to our strict definition of virtualization. This means that a large portion of the work on virtualization cannot be considered virtualization, and though potentially useful as resource allocation techniques, the energy, time and costs expended might have been better used for other research. The need for virtualization to occur within one representation level is an important lesson to bear in mind for future work, as it will hopefully lead to more meaningful research.

We hope that this thesis can play a role in increasing the awareness of virtualization theory, so that in future, more research on virtualization will be done within one representation level. If it were possible to influence standardisation, industry and regulation, one change that would be welcomed is to strictly define virtualization as recursive. This change would hopefully reduce confusion about virtualization, as it would be possible to quickly determine whether a proposed virtualization algorithm or implementation is actually virtualization.

9.3.3 Focus on WLW at the Spectrum Level

An important insight gained from the structured survey, is that wireless link virtualization has almost exclusively been done at the information level, although it is also possible at the spectrum level. There can be many benefits of virtualization at the spectrum level, such as increased flexibility and customizability, but very little work has been done on quantifying these benefits (and possible drawbacks). Without a precise understanding of virtualization, these insights would not be possible.

Noting the lack of research on spectrum-level virtualization, the second half of the thesis examined this topic in detail, and demonstrated several potential benefits and applications. It would not have been possible to achieve this without a clear definition and theory of virtualization.

9.3.4 Separating the Isolation and Embedding Problems

Focussing on a more complete knowledge of virtualization provided another advantage; the separation of the isolation and the embedding problems. Tackling these two aspects of virtualization individually, while knowing about the constraints of both problems has been very helpful at several points throughout the thesis. Firstly, it provided a better understanding of virtualization, and enables the realization that virtualization is always done in one or more dimensions of a resource. Next, when surveying the existing works, the separation of these problems provided many insights. Lastly, the separation of the isolation and embedding problems was incredibly useful when implementing wireless link virtualization practically, and allowed us to analyse each problem separately. As a result, we were able to understand these two problems better and proposed new methods of isolation and embedding.

Without first developing the theory and definition of virtualization, it would not have been possible to analyse the isolation and embedding problems as clearly. Additionally, the implementation of wireless link virtualization at the spectrum level, MySVL, might have been significantly more difficult if these two problems had been intertwined.

9.3.5 Flexible and Generic Implementation

Following on from the previous point, another benefit of the clear definition is that it allowed us to implement wireless link virtualization in a generic manner. As a result of the theory of virtualization, we recognised that existing solutions have not been many-to-many, and therefore we bore this in mind when designing our implementation, MySVL. Many-to-many virtualization can provide several advantages over many-to-one virtualization, and allows not only multiple virtual links per real link, but also multiple real links per virtual link. This is significant as the aggregation of resources can lead to new services and better spectrum efficiency and RF hardware use, such as the reuse of outdated equipment in next generation systems.

As a generic implementation of wireless link virtualization at the spectrum level, MySVL can be used with any Physical Layer (PHY) technology and parameters. Moreover,

we can use different forms of isolation, such as contiguous or non-contiguous spectrum use, depending on the scenario and the application of virtualization. However, extra measures are needed in the case of non-contiguous spectrum mapping.

Any form of embedding algorithm can be used, such as heuristic algorithms, resource auctions, optimization, or simply predefined allocation. Finally, MySVL can easily be extended to include new features, such as dynamicity, since the source code is open-source and available online. It would not have been possible to achieve such a generic implementation without the clarity provided by the theory of virtualization.

A final point concerning the generic implementation is that it allowed us to propose new applications such as using virtualization to achieve physical layer security, demonstrating reliable but cost-efficient resources for Public Protection and Disaster Relief (PPDR) services through the use of a virtual Public Safety (PS) operator, and dynamic spectrum access *without* requiring changes to the PHY layer. We also discussed additional possibilities, such as the use of virtualization to deploy new technologies alongside existing networks, and the convergence of wireless technologies.

9.4 Future Work

We briefly describe a number of research directions that result from the outcomes of this thesis, also made possible by the approach taken. There are many interesting topics to examine, but we only discuss the most important research problems.

9.4.1 Comparing Virtualization at Different Representation Levels

A very interesting research topic is to quantitatively investigate the representation level(s) that would be more suitable for virtualization for different types of wireless networks such as nomadic, mobile and ad-hoc networks. Although this question could not possibly be answered for every wireless network application, it might be possible to explore whether some applications are better suited to spectrum-level or information-level virtualization.

One consideration to bear in mind is that due to the stochastic nature of the wireless channel, it is not possible to guarantee a specific data-rate, unlike a wired connection. This could mean that spectrum-level virtualization can provide some advantages over information-level virtualization in that it allows virtual links to have customizable spectrum-level properties, such as modulation and coding scheme, the assignment of frequencies, and the choice of technology, whereas for information-level virtualization every

virtual network uses the same spectrum-level conditions and parameters. Spectrum-level virtualization would also allow spectrum resources to be used more efficiently than information-level virtualization, since wireless links can be optimized to the channel conditions and application requirements. However, the isolation required between different virtual links at the spectrum level is not as straightforward, and might require additional processing and overheads in terms of initialization and management.

Some advantages of information-level virtualization compared to spectrum level virtualization are that it might be easier and less complex. It could also be easier to integrate with wired network virtualization, since packets with Virtual Local Area Network (VLAN) tags coming from a wired virtual network can be forwarded directly from the wireless virtual access point to the nodes. Each virtual link can also easily set its own information-level settings such as data rate, latency, security and authentication, etc.

In the context of Fifth Generation Mobile Networks (5G), this research question could be very interesting, as one can imagine different scenarios in which virtualization might be useful, with different potential requirements in terms of customization, flexibility, ease-of-use, isolation and other aspects. For example, in critical future networks, such as the networks used for cyber-physical systems (e.g. industrial networks, automotive and transport networks, medical networks, and many more), very strict latency requirements, isolation, and reliability could mean that spectrum-level virtualization might be suitable, since spectrum level virtualization can offer finer customization, since each virtual network can use its own physical layer technology.

On the other hand, information-level virtualization might be more appropriate for human-oriented content delivery networks, such as office, transport, event, and home networks. The advantages of information-level virtualization could be easier/seamless integration between different networks, easier balancing of network resources and more efficient use (by dynamically slicing or aggregating resources based on the current demand), new services (such as link migration, increasing the reliability of networks), and faster development of new network technologies and services, as these could be tested on separate virtual networks without compromising existing networks.

9.4.2 Implementing Embedding with MySVL

A second potential research direction to explore would be to implement embedding techniques with MySVL, and to use them on the testbed. This would allow embedding algorithms to be tested under real conditions, and it would be possible to compare how

suitable algorithms are for different applications.

Moreover, implementing embedding algorithms on the testbed would enable the demonstration of complex and dynamic applications such as the virtual PS operator sharing a commercial network. It would be possible to measure the throughput of each user in real world conditions, the impact of the emergency scenario and the overhead of virtualization.

9.4.3 Wireless Node and Network Virtualization

In most of the wireless network virtualization work so far, the focus is on virtualizing the link resources and assuming that the node resources can be virtualized easily. However this might not always be the case. The current work on network function virtualization tackles some of the processing issues and shows that this assumption does not apply. To the best of our knowledge the only works that consider both wireless node and link virtualization are [114] and [92], however for both of these works the wireless link resource allocation does not pass the recursive test developed in Chapter 5, and thus the wireless link aspect of these works is not considered virtualization.

Building on an implementation of spectrum level link virtualization, it would be very interesting to apply some of the concepts developed in network virtualization and to create wireless virtual *networks*, rather than virtual links only. It might be possible to create wireless network virtualization implementations that allow users to dynamically request wireless virtual networks with specific, customizable requirements.

List of Acronyms

3GPP	3 rd Generation Partnership Program
5G	Fifth Generation Mobile Networks
API	Application Programming Interface
AR	Abstraction/Representation
BPSK	Binary Phase Shift Keying
CDMA	Code Division Multiple Access
CFO	Carrier Frequency Offset
CSI	Channel State Information
DSA	Dynamic Spectrum Access
DSP	Digital Signal Processing
EDI	Embedding Density Index
eNodeB	evolved Node B
ETSI	The European Telecommunications Standards Institute
FEC	Forward Error Correction
FER	Frame Error Rate
FFT	Fast Fourier Transform
FFTW	Fastest Fourier Transform in the West
GMSK	Gaussian Minimum Shift Keying

GRC	GNU Radio Companion
iFFT	Inverse Fast Fourier Transform
InP	Infrastructure Provider
IPO+S	Input-Process-Output and Storage
I/O	Input/Output
IoT	Internet of Things
LTE	Long Term Evolution
MAC	Media Access Control
MIMO	Multiple Input Multiple Output
MIP	Mixed-Integer Programming
MM	Mapping Mechanism
MVNO	Mobile Virtual Network Operator
MySVL	The Many-to-many Spectrum Virtualization Layer
NC-OFDM	Non-contiguous OFDM
NV	Network Virtualization
NIC	Network Interface Card
NFV	Network Function Virtualization
OED	Oxford English Dictionary
OFDM	Orthogonal Frequency Division Multiplexing
OFDMA	Orthogonal Frequency Division Multiple Access
OOT	Out-of-Tree
OSI	Open Systems Interconnection
PCF	Point Coordination Function
PHY	Physical Layer

PMR	Professional/Private Mobile Radio
PPDR	Public Protection and Disaster Relief
PRB	Physical Resource Block
PS	Public Safety
PSM	Power Saving Mode
PSTN	Public Switched Telephone Network
PU	Primary User
QoS	Quality of Service
QPSK	Quadrature Phase Shift Keying
RF	Radio Frequency
RL	Representation Level
RR	Real Resource
Rx	Receiver
S&C	Schmidl and Cox
SDN	Software Defined Networking
SDR	Software Defined Radio
SIMD	Single Instruction Multiple Data
SP	Service Provider
SSH	Secure Shell
SU	Secondary User
SVL	Spectrum Virtualization Layer
Tx	Transmitter
UML	Unified Modelling Language
USRP	Universal Software Radio Peripheral

VAP	Virtual Access Points
VLAN	Virtual Local Area Network
VMM	Virtual Machine Monitor
vMAC	virtual Media Access Control
VNO	Virtual Network Operator
VoIP	Voice over Internet Protocol
VPN	Virtual Private Network
VR	Virtual Resource
VRR	Virtual Resource Request
WiMAX	Worldwide Interoperability for Microwave Access
WLV	Wireless Link Virtualization
WNV	Wireless Network Virtualization
XML	eXtensible Markup Language

Bibliography

- [1] GENI concepts. [Online]. Available: <http://groups.geni.net/geni/wiki/GENIConcepts> [Last accessed: 09/28/2017]
- [2] B. Chun *et al.*, “PlanetLab: An overlay testbed for broad-coverage services,” *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 3, pp. 3–12, Jul. 2003.
- [3] Virtualisation approach: Evaluation and integration. [Online]. Available: <http://www.4ward-project.eu/index177c.html?s=publications&c=WP3> [Last accessed: 09/28/2017]
- [4] J. Pan, S. Paul, and R. Jain, “A survey of the research on future internet architectures,” *IEEE Communications Magazine*, vol. 49, no. 7, pp. 26–36, July 2011.
- [5] S. Paul, J. Pan, and R. Jain, “Architectures for the future networks and the next generation Internet: A survey,” *Computer Communications*, vol. 34, no. 1, pp. 2 – 42, 2010.
- [6] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach, “Virtual network embedding: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.
- [7] X. Wang, P. Krishnamurthy, and D. Tipper, “Wireless network virtualization,” in *2013 International Conference on Computing, Networking and Communications (ICNC)*, Jan 2013, pp. 818–822.
- [8] M. Chiosi *et al.*, “Network functions virtualisation: An introduction, benefits, enablers, challenges & call for action,” in *SDN and OpenFlow World Congress*, Oct 2012, pp. 22–24.
- [9] N. Lippis, “Network virtualization: The new building blocks of network design,” *White Paper*, 2007.
- [10] Service & System Aspects Technical Specifications Group, “3GPP technical report 32.842,” 3rd Generation Partnership Project, Tech. Rep., 2015.
- [11] P. Russell, “Understanding virtual,” in *SIGraDi 2003-[Proceedings of the 7th Iberoamerican Congress of Digital Graphics] Rosario Argentina*, Nov. 2003, pp. 5–7.
- [12] R. Shields, *The Virtual*. Routledge, 2003.

- [13] M.-S. Yoh, "The reality of virtual reality," in *Proceedings Seventh International Conference on Virtual Systems and Multimedia*, 2001, pp. 666–674.
- [14] G. Deleuze, *Difference and Repetition*. Columbia University Press, 1994, Translation by Paul Patton.
- [15] N. Negroponte, "Virtual reality: Oxymoron or pleonasm?" *Wired Magazine*, 1993.
- [16] T. K. Forde, I. Macaluso, and L. E. Doyle, "Exclusive sharing & virtualization of the cellular network," *IEEE International Symposium on Dynamic Spectrum Access Networks, DySPAN 2011*, pp. 337–348, 2011.
- [17] J. Sahoo, S. Mohapatra, and R. Lath, "Virtualization: A survey on concepts, taxonomy and associated security issues," in *Computer and Network Technology (ICCNT), 2010 Second International Conference on*, Apr. 2010, pp. 222–226.
- [18] H. Wen, P. K. Tiwary, and T. Le-Ngoc, "Current trends and perspectives in wireless virtualization," *International Conference on Selected Topics in Mobile and Wireless Networking, MoWNeT 2013*, pp. 62–67, 2013.
- [19] C. Liang and F. R. Yu, "Wireless network virtualization: A survey, some research issues and challenges," *Communications Surveys & Tutorials, IEEE*, vol. 17, no. 1, pp. 358–380, 2014.
- [20] —, "Wireless virtualization for next generation mobile cellular networks," *IEEE Wireless Communications*, no. 1, pp. 61–69, Feb. 2015.
- [21] J. Sachs and S. Baucke, "Virtual radio: a framework for configurable radio networks," *Proceedings of the 4th International ICST Conference on Wireless Internet*, pp. 61–68, 2008.
- [22] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal, "NFV: State of the art, challenges, and implementation in next generation mobile networks (vEPC)," *IEEE Network*, vol. 28, no. 6, pp. 18–26, Nov 2014.
- [23] D. C. Horsman, "Abstraction/Representation theory for heterotic physical computing," *Philosophical Transactions of the Royal Society of London A*, vol. 373, no. 2046, 2015.
- [24] C. Horsman, S. Stepney, R. C. Wagner, and V. Kendon, "When does a physical system compute?" in *Proceedings of the Royal Society of London A*, vol. 470, no. 2169, 2014.
- [25] D. Schmandt-Besserat, "The earliest precursor of writing," *Scientific American*, vol. 238, no. 6, pp. 50–59, 1978.
- [26] D. H. Krantz, P. Suppes, and R. D. Luce, *Foundations of measurement, Vol. I: Additive and polynomial representations*. Academic Press, 1971.
- [27] Data in the computer. [Online]. Available: http://homepage.cs.uri.edu/book/binary_data/binary_data.htm [Last accessed: 09/28/2017]

- [28] A. S. Tanenbaum and J. R. Goodman, *Structured Computer Organization*, 4th ed. Prentice Hall PTR, 1998.
- [29] R. P. Goldberg, “Architecture of virtual machines,” in *Proceedings of the Workshop on Virtual Computer Systems*, 1973, pp. 74–112.
- [30] P. J. Denning, “Virtual memory,” *ACM Comput. Surv.*, vol. 2, no. 3, pp. 153–189, 1970.
- [31] G. J. Popek and R. P. Goldberg, “Formal requirements for virtualizable third generation architectures,” *Communications of the ACM*, vol. 17, no. 7, pp. 412–421, Jul. 1974.
- [32] G. Belpaire and N.-T. Hsu, “Formal properties of recursive virtual machine architectures,” in *Proceedings of the Fifth ACM Symposium on Operating Systems Principles*, 1975, pp. 89–96.
- [33] K.-M. Park and C.-K. Kim, “A framework for virtual network embedding in wireless networks,” in *Proceedings of the 4th International Conference on Future Internet Technologies*, 2009, pp. 5–7.
- [34] T. Kilburn, D. B. Edwards, M. Lanigan, and F. H. Sumner, “One-level storage system,” *Electronic Computers, IRE Transactions on*, vol. 11, no. 2, pp. 223–235, 1962.
- [35] J. Fotheringham, “Dynamic storage allocation in the Atlas computer, including an automatic use of a backing store,” *Communications of the ACM*, vol. 4, no. 10, pp. 435–436, 1961.
- [36] H. C. Lauer and D. Wyeth, “A recursive virtual machine architecture,” in *Proceedings of the Workshop on Virtual Computer Systems*. New York, NY, USA: ACM, 1973, pp. 113–116.
- [37] P. Checkland, *Systems Thinking, Systems Practice*. John Wiley and Sons, 1981.
- [38] F. J. Corbató, M. Merwin-daggett, and R. C. Daley, “An experimental time-sharing system,” in *AIEE-IRE '62 (Spring)*, 1962, pp. 335–344.
- [39] R. J. Creasy, “The origin of the VM/370 time-sharing system,” *IBM Journal of Research and Development*, vol. 25, no. 5, pp. 483–790, 1981.
- [40] Bob Bremer. Origins of timesharing. [Online]. Available: <http://www.bobbemer.com/TIMESHAR.HTM> [Last accessed: 09/28/2017]
- [41] J. M. McKinney, “A survey of analytical time-sharing models,” *ACM Comput. Surv.*, vol. 1, no. 2, pp. 105–116, Jun. 1969.
- [42] L. Kleinrock, “A continuum of time-sharing scheduling algorithms,” in *Spring Joint Computer Conference, Proceedings of the*, 1970, pp. 453–458.

- [43] S. F. Yashkov and A. S. Yashkova, "Processor sharing: A survey of the mathematical theory," *Automation and Remote Control*, vol. 68, no. 9, pp. 1662–1731, 2007.
- [44] Timesharing: A solution to computer bottlenecks. [Online]. Available: <https://www.youtube.com/watch?v=Q07PhW5sCEk> [Last accessed: 09/28/2017]
- [45] D. Sayre, "Is automatic "folding" of programs efficient enough to displace manual?" *Communications of the ACM*, vol. 12, no. 12, pp. 656–660, 1969.
- [46] S. Raina, "Virtual shared memory: A survey of techniques and systems," *University of Bristol, Bristol, UK*, 1992.
- [47] J. Protic, M. Tomasevic, and V. Milutinović, *Distributed Shared Memory: Concepts and Systems*. John Wiley & Sons, 1998, vol. 21.
- [48] F. J. Corbató and V. A. Vyssotsky, "Introduction and overview of the Multics system," in *Proceedings of the 1965 Fall Joint Computer Conference, Part I*, 1965, pp. 185–196.
- [49] A. J. Critcklow, "Generalized multiprocessing and multiprogramming systems," in *Proceedings of the November 12-14, 1963, Fall Joint Computer Conference*, 1963, pp. 107–126.
- [50] J. P. Buzen and U. O. Gagliardi, "The evolution of virtual machine architecture," *Proceedings of the June 48 1973 national computer conference and exposition on AFIPS 73*, pp. 291–299, 1973.
- [51] R. P. Goldberg, "Architectural principles for virtual computer systems," DTIC Document, Tech. Rep., 1973.
- [52] —, "Survey of virtual machine research," *Computer*, vol. 7, no. 6, pp. 34–45, Jun. 1974.
- [53] I. Ahmad, J. Anderson, A. Holler, R. Kambo, and V. Makhija, "An analysis of disk performance in VMware ESX server virtual machines," in *Workload Characterization, 2003. WWC-6. 2003 IEEE International Workshop on*, Oct. 2003, pp. 65–76.
- [54] R. Rose, "Survey of system virtualization techniques," Oregon State University, Tech. Rep., 2004.
- [55] M. F. Bari *et al.*, "Data center network virtualization: A survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 2, pp. 909–928, Feb. 2013.
- [56] S. N. T.-c. Chiueh and S. Brook, "A survey on virtualization technologies," *RPE Report*, pp. 1–42, 2005.
- [57] A. Blenk, A. Basta, M. Reisslein, and W. Kellerer, "Survey on network virtualization hypervisors for software defined networking," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 655–685, First Quarter 2016.

- [58] D. Abramson, "Intel virtualization technology for directed I/O," *Intel Technology Journal*, vol. 10, no. 3, pp. 179–192, 2006.
- [59] B. Zhang *et al.*, "A survey on I/O virtualization and optimization," in *ChinaGrid Conference (ChinaGrid), 2010 Fifth Annual*, Jul. 2010, pp. 117–123.
- [60] A. Rybczynski, "X.25 interface and end-to-end virtual circuit service characteristics," *IEEE Transactions on Communications*, vol. 28, no. 4, pp. 500–510, Apr. 1980.
- [61] C. Metz, "The latest in virtual private networks: Part I," *IEEE Internet Computing*, vol. 7, no. 1, pp. 87–91, Jan. 2003.
- [62] J. Carapinha, J. Jiménez, and C. E. Vargas, "Network virtualization - a view from the bottom," *Business*, pp. 73–80, 2009.
- [63] N. Chowdhury and R. Boutaba, "Network virtualization: State of the art and research challenges," *Communications Magazine, IEEE*, vol. 47, no. 7, pp. 20–26, Jul. 2009.
- [64] R. Nejabati, E. Escalona, S. Peng, and D. Simeonidou, "Optical network virtualization," in *15th International Conference on Optical Network Design and Modeling - ONDM 2011*, Feb 2011, pp. 1–5.
- [65] R. Jain and S. Paul, "Network virtualization and software defined networking for cloud computing: a survey," *IEEE Communications Magazine*, vol. 51, no. 11, pp. 24–31, November 2013.
- [66] L. Peterson, T. Anderson, D. Culler, and T. Roscoe, "A blueprint for introducing disruptive technology into the Internet," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 1, pp. 59–64, 2003.
- [67] J. S. Turner and D. E. Taylor, "Diversifying the Internet," in *GLOBECOM '05. IEEE Global Telecommunications Conference, 2005.*, vol. 2, Dec. 2005, pp. 6 pp.–760.
- [68] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through virtualization," *Computer*, vol. 38, no. 4, pp. 34–41, 2005.
- [69] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles*, 2001, pp. 131–145.
- [70] J. Touch, Y. Wang, L. Eggert, and G. Finn, "A virtual internet architecture," *ISI Technical Report ISI-TR-2003-570*, pp. 73–80, 2003.
- [71] N. Feamster, L. Gao, and J. Rexford, "How to lease the Internet in your spare time," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 1, pp. 61–64, 2007.

- [72] N. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Computer Networks*, vol. 54, no. 5, pp. 862 – 876, 2010.
- [73] B. A. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1617–1634, Third 2014.
- [74] D. Kreutz *et al.*, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan 2015.
- [75] N. W. Paper, "Network functions virtualisation: An introduction, benefits, enablers, challenges & call for action. issue 1," Oct. 2012.
- [76] R. Mijumbi *et al.*, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2015.
- [77] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90–97, Feb 2015.
- [78] R. Chandra and P. Bahl, "MultiNet: Connecting to multiple IEEE 802.11 networks using a single wireless card," in *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, 2004, pp. 882–893.
- [79] M. Yang *et al.*, "Software-defined and virtualized future mobile and wireless networks: A survey," *Mobile Networks and Applications*, vol. 20, no. 1, pp. 4–18, 2015.
- [80] A. Wang, M. Iyer, R. Dutta, G. N. Rouskas, and I. Baldine, "Network virtualization: Technologies, perspectives, and frontiers," *Journal of Lightwave Technology*, vol. 31, no. 4, pp. 523–537, 2013.
- [81] N. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *INFOCOM 2009, IEEE*, 2009, pp. 783–791.
- [82] X. Lu, K. Yang, Y. Liu, D. Zhou, and S. Liu, "An elastic resource allocation algorithm enabling wireless network virtualization," *Wireless Communications and Mobile Computing*, vol. 15, no. 2, pp. 295–308, 2015.
- [83] P. Lv, X. Wang, and M. Xu, "Virtual access network embedding in wireless mesh networks," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1362 – 1378, 2012.
- [84] X. Xu, D. Wang, X. Tao, and T. Svensson, "Resource pooling for frameless network architecture with adaptive resource allocation," *Science China Information Sciences*, vol. 56, no. 2, pp. 1–12, 2013.

- [85] X. Xu, H. Zhang, Y. Hou, X. Tao, and P. Zhang, "SDN based next generation mobile network with service slicing and trials," *China Communications*, vol. 11, no. 2, pp. 65–77, Feb. 2014.
- [86] G. Liu, F. R. Yu, H. Ji, and V. C. M. Leung, "Distributed resource allocation in full-duplex relaying networks with wireless virtualization," in *Global Communications Conference (GLOBECOM), 2014 IEEE*, Dec. 2014, pp. 4959–4964.
- [87] G. Liu, F. R. Yu, H. Ji, V. C. Leung, and X. Li, "In-band full-duplex relaying: A survey, research issues and challenges," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 500–524, 2015.
- [88] M. Kamel, L. Bao Le, and A. Girard, "LTE multi-cell dynamic resource allocation for wireless network virtualization," in *Wireless Communications and Networking Conference (WCNC), 2015 IEEE*, Mar. 2015, pp. 966–971.
- [89] G. Zhang, K. Yang, J. Wei, K. Xu, and P. Liu, "Virtual resource allocation for wireless virtualization networks using market equilibrium theory," in *Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on*, Apr. 2015, pp. 366–371.
- [90] G. Zhang, K. Yang, K. Xu, and Y. Dong, "A bargaining game theoretic method for virtual resource allocation in LTE-based cellular networks," *Science China Information Sciences*, vol. 58, no. 12, pp. 1–9, 2015.
- [91] S. Khatibi and L. M. Correia, "A model for virtual radio resource management in virtual RANs," *EURASIP Journal on Wireless Communications and Networking*, vol. 2015, no. 1, p. 68, 2015.
- [92] M. M. Rahman, C. Despins, and S. Affes, "Design optimization of wireless access virtualization based on cost QoS trade-off utility maximization," *IEEE Transactions on Wireless Communications*, vol. 15, no. 9, pp. 6146–6162, Sept 2016.
- [93] Z. Feng *et al.*, "An effective approach to 5G: Wireless network virtualization," *IEEE Communications Magazine*, vol. 53, no. 12, pp. 53–59, Dec. 2015.
- [94] Y. Cai, F. R. Yu, C. Liang, B. Sun, and Q. Yan, "Software defined device-to-device (D2D) communications in virtual wireless networks with imperfect network state information (NSI)," *IEEE Transactions on Vehicular Technology*, vol. PP, no. 99, pp. 1–1, 2015.
- [95] L. Chen, F. R. Yu, H. Ji, G. Liu, and V. Leung, "Distributed virtual resource allocation in small cell networks with full duplex self-backhauls and virtualization," *IEEE Transactions on Vehicular Technology*, vol. PP, no. 99, pp. 1–1, 2015.
- [96] C. Liang and F. R. Yu, "Distributed resource allocation in virtualized wireless cellular networks based on ADMM," in *Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on*, Apr. 2015, pp. 360–365.
- [97] F. Fu and U. C. Kozat, "Wireless network virtualization as a sequential auction game," *Proceedings - IEEE INFOCOM*, pp. 1–9, 2010.

- [98] F. Fu and U. Kozat, “Stochastic game for wireless network virtualization,” *Networking, IEEE/ACM Transactions on*, vol. 21, no. 1, pp. 84–97, 2012.
- [99] M. I. Kamel, L. B. Le, and A. Girard, “LTE wireless network virtualization: Dynamic slicing via flexible scheduling,” in *Vehicular Technology Conference (VTC Fall), 2014 IEEE 80th*, Sep. 2014, pp. 1–5.
- [100] Y. Zaki, L. Zhao, C. Görg, and A. Timm-Giel, “LTE mobile network virtualization exploiting multiplexing and multi-user diversity gain,” *Mobile Networks and Applications*, vol. 16, no. 4, pp. 424–432, 2011.
- [101] —, *Mobile Networks and Management: Second International ICST Conference, MONAMI 2010, Santander, Spain, September 22-24, 2010, Revised Selected Papers*. Springer, 2011, ch. A Novel LTE Wireless Virtualization Framework, pp. 245–257.
- [102] L. Zhao, M. Li, Y. Zaki, A. Timm-Giel, and C. Görg, “LTE virtualization: From theoretical gain to practical solution,” in *Teletraffic Congress (ITC), 2011 23rd International*, Sep. 2011, pp. 71–78.
- [103] M. Li *et al.*, “Investigation of network virtualization and load balancing techniques in LTE networks,” in *Vehicular Technology Conference (VTC Spring), 2012 IEEE 75th*, May 2012, pp. 1–5.
- [104] B. Liu and H. Tian, “A bankruptcy game-based resource allocation approach among virtual mobile operators,” *Communications Letters, IEEE*, vol. 17, no. 7, pp. 1420–1423, 2013.
- [105] J. S. Panchal, R. D. Yates, and M. M. Buddhikot, “Mobile network resource sharing options: Performance comparisons,” *IEEE Transactions on Wireless Communications*, vol. 12, no. 9, pp. 4470–4482, Sep. 2013.
- [106] M. Kalil, A. Shami, and Y. Ye, “Wireless resources virtualization in LTE systems,” in *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, 2014, pp. 363–368.
- [107] M. Kalil, A. Moubayed, A. Shami, and A. Al-Dweik, “Efficient low-complexity scheduler for wireless resource virtualization,” *IEEE Wireless Communications Letters*, vol. 5, no. 1, pp. 56–59, Feb 2016.
- [108] A. Moubayed, A. Shami, and H. Lutfiyya, “Wireless resource virtualization with device-to-device communication underlaying LTE network,” *IEEE Transactions on Broadcasting*, vol. 61, no. 4, pp. 734–740, Dec 2015.
- [109] R. Kokku, R. Mahindra, H. Zhang, and S. Rangarajan, “NVS: A substrate for virtualizing wireless resources in cellular networks,” *Networking, IEEE/ACM Transactions on*, vol. 20, no. 5, pp. 1333–1346, 2012.
- [110] —, “CellSlice: Cellular wireless resource slicing for active RAN sharing,” in *Communication Systems and Networks (COMSNETS), 2013 Fifth International Conference on*, 2013, pp. 1–10.

- [111] X. Costa-Perez, J. Swetina, T. Guo, R. Mahindra, and S. Rangarajan, "Radio access network virtualization for future mobile carrier networks," *Communications Magazine, IEEE*, vol. 51, no. 7, pp. 27–35, 2013.
- [112] J. Wei, K. Yang, G. Zhang, and Z. Hu, "Pricing-based power allocation in wireless network virtualization: A game approach," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2015 International*, Aug. 2015, pp. 188–193.
- [113] X. Wang, P. Krishnamurthy, and D. Tipper, "A collaborative spectrum-sharing framework for LTE virtualization," in *2015 IEEE Conference on Collaboration and Internet Computing (CIC)*, Oct 2015, pp. 260–269.
- [114] G. D. Stasi, S. Avallone, and R. Canonico, "Virtual network embedding in wireless mesh networks through reconfiguration of channels," in *2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct 2013, pp. 537–544.
- [115] Q. Zhu and X. Zhang, "Game-theory based power and spectrum virtualization for maximizing spectrum efficiency over mobile cloud-computing wireless networks," in *Information Sciences and Systems (CISS), 2015 49th Annual Conference on*, Mar. 2015, pp. 1–6.
- [116] —, "Bayesian-game based power and spectrum virtualization for maximizing spectrum efficiency over mobile cloud-computing wireless networks," in *Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on*, Apr. 2015, pp. 378–383.
- [117] B. Fan, H. Tian, and B. Liu, "Game theory based power allocation in LTE air interface virtualization," in *2015 IEEE Wireless Communications and Networking Conference (WCNC)*, March 2015, pp. 972–976.
- [118] K. Zhu and E. Hossain, "Virtualization of 5G cellular networks as a hierarchical combinatorial auction," *IEEE Transactions on Mobile Computing*, vol. 15, no. 10, pp. 2640–2654, Oct 2016.
- [119] H. Ahmadi, I. Macaluso, I. Gomez, L. Doyle, and L. DaSilva, "Substitutability of spectrum and cloud-based antennas in virtualised wireless networks," *IEEE Wireless Communications*, 2016.
- [120] H. Ahmadi, I. Macaluso, I. Gomez, L. DaSilva, and L. Doyle, "Virtualization of spatial streams for enhanced spectrum sharing," in *Global Telecommunications Conference (GLOBECOM 2016), 2016 IEEE*, 2016.
- [121] N. Bhushan *et al.*, "Network densification: the dominant theme for wireless evolution into 5G," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 82–89, February 2014.
- [122] M. Yang *et al.*, "Opportunistic spectrum sharing based resource allocation for wireless virtualization," in *Innovative Mobile and Internet Services in Ubiquitous*

- Computing (IMIS), 2013 Seventh International Conference on.* IEEE, Jul. 2013, pp. 51–58.
- [123] S. Paul and S. Seshan, “Technical document on wireless virtualization,” *GENI: Global Environment for Network Innovations,, Tech. Rep*, Sep. 2006.
 - [124] K. Tan, H. Shen, J. Zhang, and Y. Zhang, “Enable flexible spectrum access with spectrum virtualization,” in *Dynamic Spectrum Access Networks (DYSPAN), 2012 IEEE International Symposium on*, 2012, pp. 47–58.
 - [125] M. Yang *et al.*, “Opportunistic sharing scheme for spectrum allocation in wireless virtualization,” *Soft Computing*, vol. 18, no. 9, pp. 1685–1696, 2014.
 - [126] M. Yang, Y. Li, L. Zeng, D. Jin, and L. Su, “Karnaugh-map like online embedding algorithm of wireless virtualization,” in *Wireless Personal Multimedia Communications (WPMC), 2012 15th International Symposium on*, 2012, pp. 594–598.
 - [127] J. van de Belt, H. Ahmadi, and L. Doyle, “A dynamic embedding algorithm for wireless network virtualization,” in *Vehicular Technology Conference (VTC Fall), 2014 IEEE 80th*, Sep. 2014, pp. 1–6.
 - [128] J. van de Belt, H. Ahmadi, L. Doyle, and O. Sallent, “A prioritised traffic embedding mechanism enabling a public safety virtual operator,” in *Vehicular Technology Conference (VTC Fall), 2015 IEEE 82nd*, Sep. 2015, pp. 1–5.
 - [129] F.-T. Hsu and C.-H. Gan, “Resource allocation with spectrum aggregation for wireless virtual network embedding,” in *Vehicular Technology Conference (VTC Fall), 2015 IEEE 82nd*, Sep. 2015, pp. 1–5.
 - [130] G. Lu, C. Liu, L. Li, and Q. Yuan, “A dynamic allocation algorithm for physical carrier resource in BBU pool of virtualized wireless network,” in *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2015 International Conference on*, Sep. 2015, pp. 434–441.
 - [131] X. Zhang *et al.*, “Efficient resource allocation for wireless virtualization using time-space division,” in *2012 8th International Wireless Communications and Mobile Computing Conference (IWCMC)*, Aug 2012, pp. 59–64.
 - [132] D. Raychaudhuri *et al.*, “Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols,” in *Wireless Communications and Networking Conference, 2005 IEEE*, vol. 3, Mar. 2005, pp. 1664–1669.
 - [133] G. Smith, A. Chaturvedi, A. Mishra, and S. Banerjee, “Wireless virtualization on commodity 802.11 hardware,” in *Proceedings of the Second ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization*, 2007, pp. 75–82.
 - [134] H. Coskun, I. Schieferdecker, and Y. Al-Hazmi, “Virtual WLAN: Going beyond virtual access points,” *Electronic Communications of the EASST*, vol. 17, 2009.

- [135] Y. Al-Hazmi and H. de Meer, "Virtualization of 802.11 interfaces for wireless mesh networks," in *Wireless On-Demand Network Systems and Services (WONS), 2011 Eighth International Conference on*, Jan. 2011, pp. 44–51.
- [136] P. Lv, Z. Cai, J. Xu, and M. Xu, "Multicast service-oriented virtual network embedding in wireless mesh networks," *IEEE Communications Letters*, vol. 16, no. 3, pp. 375–377, March 2012.
- [137] R. Mahindra *et al.*, "Space versus time separation for wireless virtualization on an indoor grid," in *Next Generation Internet Networks, 2008*, Apr. 2008, pp. 215–222.
- [138] S. Perez, J. M. Cabero, and E. Miguel, "Virtualization of the wireless medium: A simulation-based study," in *Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th*, Apr. 2009, pp. 1–5.
- [139] R. Sherwood *et al.*, "Flowvisor: A network virtualization layer," *OpenFlow Switch Consortium, Tech. Rep.*, pp. 1–13, 2009.
- [140] G. Bhanage, D. Vete, I. Seskar, and D. Raychaudhuri, "SplitAP: Leveraging wireless network virtualization for flexible sharing of WLANs," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, Dec. 2010, pp. 1–6.
- [141] G. Bhanage, I. Seskar, R. Mahindra, and D. Raychaudhuri, "Virtual basestation: Architecture for an open shared WiMAX framework," in *Proceedings of the Second ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures*, 2010, pp. 1–8.
- [142] L. Xia, S. Kumar, X. Yang, and P. Gopalakrishnan, "Virtual WiFi: Bring virtualization from wired to wireless," *ACM SIGPLAN Notices*, vol. 46, no. 7, pp. 181–192, 2011.
- [143] G. Aljabari and E. Eren, "Virtualization of wireless LAN infrastructures," in *Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), 2011 IEEE 6th International Conference on*, vol. 2, Sep. 2011, pp. 837–841.
- [144] K. Nakauchi, Y. Shoji, and N. Nishinaga, "Airtime-based resource control in wireless LANs for wireless network virtualization," in *Ubiquitous and Future Networks (ICUFN), 2012 Fourth International Conference on*, Jul. 2012, pp. 166–169.
- [145] K. Katsalis *et al.*, "Wireless network virtualization: The CONTENT project approach," in *Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2014 IEEE 19th International Workshop on*, Dec. 2014, pp. 90–94.
- [146] T. Spyropoulos, S. Fdida, and S. Kirkpatrick, "Future internet: Fundamentals and measurement," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 2, pp. 101–106, 2007.
- [147] B. Aboba, "Virtual access points," IEEE, Tech. Rep., Mar. 2003.
- [148] H. Wen, P. K. Tiwary, and T. Le-Ngoc, *Wireless Virtualization*. Springer, 2014.

- [149] M. Richart, J. Baliosian, J. Serrat, and J.-L. Gorricho, "Resource slicing in virtual wireless networks: A survey," *IEEE Transactions on Network and Service Management*, 2016.
- [150] M. Yang *et al.*, "OpenRAN: A software-defined ran architecture via virtualization," *ACM SIGCOMM computer communication review*, vol. 43, no. 4, pp. 549–550, Aug. 2013.
- [151] R. Jordan and C. T. Abdallah, "Wireless communications and networking: an overview," *IEEE Antennas and Propagation Magazine*, vol. 44, no. 1, pp. 185–193, 2002.
- [152] Z. Pi and F. Khan, "An introduction to millimeter-wave mobile broadband systems," *IEEE Communications Magazine*, vol. 49, no. 6, 2011.
- [153] T. S. Rappaport, S. Sun, R. Mayzus, H. Zhao, Y. Azar, K. Wang, G. N. Wong, J. K. Schulz, M. Samimi, and F. Gutierrez, "Millimeter wave mobile communications for 5G cellular: It will work!" *IEEE access*, vol. 1, pp. 335–349, 2013.
- [154] M. Rebato, F. Boccardi, M. Mezzavilla, S. Rangan, and M. Zorzi, "Hybrid spectrum access for mmwave networks," in *2016 Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, June 2016, pp. 1–7.
- [155] R. Zhang, M. Wang, L. X. Cai, Z. Zheng, X. Shen, and L. L. Xie, "LTE-unlicensed: the future of spectrum aggregation for cellular networks," *IEEE Wireless Communications*, vol. 22, no. 3, pp. 150–159, June 2015.
- [156] W. Wang, Z. Zhang, and A. Huang, "Spectrum aggregation: Overview and challenges," *Network Protocols and Algorithms*, vol. 2, no. 1, pp. 184–196, 2010.
- [157] H. Lee, S. Vahid, and K. Moessner, "A survey of radio resource management for spectrum aggregation in LTE-advanced," *IEEE Communications Surveys Tutorials*, vol. 16, no. 2, pp. 745–760, Second 2014.
- [158] Windows 7 adds virtual Wifi technology from Microsoft research. [Online]. Available: <http://www.istartedsomething.com/20090516/windows-7-native-virtual-wifi-technology-microsoft-research/> [Last accessed: 09/28/2017]
- [159] W. Qu and S. Srinivas, "IPSec-based secure wireless virtual private network," in *MILCOM 2002. Proceedings*, vol. 2, Oct 2002, pp. 1107–1112 vol.2.
- [160] P. J. Weiser, *The Untapped Promise of Wireless Spectrum*. The Brookings Institution, Jul. 2008.
- [161] S. Yun, D. Kim, and L. Qiu, "Fine-grained spectrum adaptation in WiFi networks," in *Proceedings of the 19th Annual International Conference on Mobile Computing & Networking*, ser. MobiCom '13. New York, NY, USA: ACM, 2013, pp. 327–338.

- [162] E. Chai, K. G. Shin, J. Lee, S. J. Lee, and R. H. Etkin, “Fast spectrum shaping for next-generation wireless networks,” *IEEE Transactions on Mobile Computing*, vol. 13, no. 1, pp. 20–34, Jan 2014.
- [163] W. Wang *et al.*, “Changing channel without strings: Coordination-free wideband spectrum adaptation,” in *2015 IEEE Conference on Computer Communications (INFOCOM)*, April 2015, pp. 460–468.
- [164] W. Wang, Y. Chen, Q. Zhang, and T. Jiang, “A software-defined wireless networking enabled spectrum management architecture,” *IEEE Communications Magazine*, vol. 54, no. 1, pp. 33–39, January 2016.
- [165] S. S. Hong, J. Mehlman, and S. Katti, “Picasso: Flexible RF and spectrum slicing,” *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 37–48, 2012.
- [166] S. Yun and L. Qiu, “Supporting WiFi and LTE co-existence,” in *2015 IEEE Conference on Computer Communications (INFOCOM)*, April 2015, pp. 810–818.
- [167] B. Chen, V. Yenamandra, and K. Srinivasan, “FlexRadio: Fully flexible radios and networks.” in *NSDI*, 2015, pp. 205–218.
- [168] K. Tan *et al.*, “Sora: High performance software radio using general purpose multi-core processors,” in *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*. USENIX Association, 2009, pp. 75–90.
- [169] Sora high performance software radio is now open source. [Online]. Available: <https://www.microsoft.com/en-us/research/blog/sora-high-performance-software-radio-is-now-open-source/> [Last accessed: 09/28/2017]
- [170] Microsoft research software radio (Sora). [Online]. Available: <https://www.microsoft.com/en-us/research/project/microsoft-research-software-radio-sora/> [Last accessed: 09/28/2017]
- [171] J. Mitola, “The software radio architecture,” *IEEE Communications Magazine*, vol. 33, no. 5, pp. 26–38, May 1995.
- [172] E. Blossom, “GNU radio: Tools for exploring the radio frequency spectrum,” *Linux J.*, vol. 2004, no. 122, pp. 4–, Jun. 2004.
- [173] Guided tutorials. [Online]. Available: https://wiki.gnuradio.org/index.php/Guided_Tutorials [Last accessed: 09/28/2017]
- [174] OutfTreemodules – GNU radio.
- [175] The comprehensive GNU radio archive network. [Online]. Available: <http://www.cgran.org/> [Last accessed: 09/28/2017]
- [176] GNU radio companion. [Online]. Available: <https://wiki.gnuradio.org/index.php/GNURadioCompanion> [Last accessed: 09/28/2017]

- [177] Guided tutorial GRC. [Online]. Available: https://wiki.gnuradio.org/index.php/Guided_Tutorial_GRC [Last accessed: 09/28/2017]
- [178] P. D. Sutton *et al.*, “Iris: an architecture for cognitive radio networking testbeds,” *IEEE Communications Magazine*, vol. 48, no. 9, pp. 114–122, Sept 2010.
- [179] P. Pawelczak, K. Nolan, L. Doyle, S. W. Oh, and D. Cabric, “Cognitive radio: Ten years of experimentation and development,” *IEEE Communications Magazine*, vol. 49, no. 3, pp. 90–100, March 2011.
- [180] LuaRadio. [Online]. Available: <http://luaradio.io/> [Last accessed: 09/28/2017]
- [181] J. Snyder, B. McNair, S. Edwards, and C. Dietrich, “OSSIE: An open source software defined radio platform for education and research,” in *International conference on frontiers in education: computer science and computer engineering (FECS’11). World congress in computer science, Computer engineering and applied computing. Las Vegas, NV*, 2011.
- [182] Soundcard SDR basics, SDR sharp. [Online]. Available: <https://sites.google.com/site/g4zfqradio/connecting-soundcard-sdr-to-computer> [Last accessed: 09/28/2017]
- [183] High definition software defined radio. [Online]. Available: <http://hdsdr.de> [Last accessed: 09/28/2017]
- [184] J. Rumbaugh, I. Jacobson, and G. Booch, *The Unified Modeling Language Reference Manual*, 2nd ed. Addison-Wesley Professional, 2010.
- [185] mutex – C++ reference. [Online]. Available: <http://www.cplusplus.com/reference/mutex/mutex/> [Last accessed: 09/28/2017]
- [186] GNU radio manual and C++ API reference: gr::block class reference. [Online]. Available: https://gnuradio.org/doc/doxygen/classgr_1_1block.html [Last accessed: 09/28/2017]
- [187] What is VPN Cascading. [Online]. Available: <https://www.le-vpn.com/what-is-vpn-cascading/> [Last accessed: 09/28/2017]
- [188] M. Song, C. Xin, Y. Zhao, and X. Cheng, “Dynamic spectrum access: from cognitive radio to network radio,” *IEEE Wireless Communications*, vol. 19, no. 1, pp. 23–29, February 2012.
- [189] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty, “NeXt generation/dynamic spectrum access/cognitive radio wireless networks: A survey,” *Computer Networks*, vol. 50, no. 13, pp. 2127 – 2159, 2006.
- [190] Q. Zhao and B. M. Sadler, “A survey of dynamic spectrum access,” *IEEE signal processing magazine*, vol. 24, no. 3, pp. 79–89, 2007.
- [191] N200/N210 Ettus knowledge base. [Online]. Available: <https://kb.ettus.com/N200/N210> [Last accessed: 09/28/2017]

- [192] X300/X310 Ettus knowledge base. [Online]. Available: <https://kb.ettus.com/X300/X310> [Last accessed: 09/28/2017]
- [193] Products— SRS. [Online]. Available: <http://www.softwareradiosystems.com/products/#srslte> [Last accessed: 09/28/2017]
- [194] CONNECT's testbed. [Online]. Available: <https://iris-testbed.connectcentre.ie/> [Last accessed: 09/28/2017]
- [195] General documentation. [Online]. Available: https://iris-testbed.connectcentre.ie/reservation/general_documentation.doc [Last accessed: 09/28/2017]
- [196] OpenNebula – flexible enterprise cloud made simple. [Online]. Available: <https://opennebula.org/> [Last accessed: 09/28/2017]
- [197] E. Dahlman, S. Parkvall, and J. Skold, *4G: LTE/LTE-advanced for Mobile Broadband*. Academic press, 2013.
- [198] T. M. Schmidl and D. C. Cox, "Robust frequency and timing synchronization for OFDM," *IEEE Transactions on Communications*, vol. 45, no. 12, pp. 1613–1621, 1997.
- [199] S. E. Kocabas and A. Atalar, "Binary sequences with low aperiodic autocorrelation for synchronization purposes," *IEEE Communications Letters*, vol. 7, no. 1, pp. 36–38, 2003.
- [200] GNU radio manual and C++ API reference: Stream tags. [Online]. Available: https://gnuradio.org/doc/doxygen/page_stream_tags.html [Last accessed: 09/28/2017]
- [201] D. Gottlieb and C.-W. Shu, "On the Gibbs phenomenon and its resolution," *SIAM review*, vol. 39, no. 4, pp. 644–668, 1997.
- [202] G.-C. Hsieh and J. C. Hung, "Phase-locked loop techniques. a survey," *IEEE Transactions on Industrial Electronics*, vol. 43, no. 6, pp. 609–615, Dec 1996.
- [203] M. Morelli, C. C. J. Kuo, and M. O. Pun, "Synchronization techniques for orthogonal frequency division multiple access (OFDMA): A tutorial review," *Proceedings of the IEEE*, vol. 95, no. 7, pp. 1394–1427, July 2007.
- [204] S. Mohseni and M. A. Matin, "Study of the estimation techniques for the carrier frequency offset (CFO) in OFDM systems," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 12, no. 6, p. 73, 2012.
- [205] J. P. Costas, "Synchronous communications," *Proceedings of the IRE*, vol. 44, no. 12, pp. 1713–1718, Dec 1956.
- [206] UHD and USRP Manual. [Online]. Available: https://files.ettus.com/manual/page_general.html/ [Last accessed: 09/28/2017]

- [207] R. Chávez-Santiago, M. Szydelko, A. Kliks, F. Foukalas, Y. Haddad, K. E. Nolan, M. Y. Kelly, M. T. Masonta, and I. Balasingham, “5G: The convergence of wireless communications,” *Wireless Personal Communications*, vol. 83, no. 3, pp. 1617–1642, Aug 2015.
- [208] T. Yucek and H. Arslan, “A survey of spectrum sensing algorithms for cognitive radio applications,” *IEEE Communications Surveys Tutorials*, vol. 11, no. 1, pp. 116–130, First 2009.
- [209] R. Rajbanshi, A. M. Wyglinski, and G. J. Minden, “An efficient implementation of NC-OFDM transceivers for cognitive radios,” in *2006 1st International Conference on Cognitive Radio Oriented Wireless Networks and Communications*, June 2006, pp. 1–5.
- [210] L. Dong, Z. Han, A. P. Petropulu, and H. V. Poor, “Improving wireless physical layer security via cooperating relays,” *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1875–1888, March 2010.
- [211] N. Cai and T. Chan, “Theory of secure network coding,” *Proceedings of the IEEE*, vol. 99, no. 3, pp. 421–437, March 2011.
- [212] F. Meucci, S. A. Wardana, and N. R. Prasad, “Secure physical layer using dynamic permutations in cognitive OFDMA systems,” in *VTC Spring 2009 - IEEE 69th Vehicular Technology Conference*, April 2009, pp. 1–5.
- [213] H. Dyckhoff, “A typology of cutting and packing problems,” *European Journal of Operational Research*, vol. 44, no. 2, pp. 145–159, 1990.
- [214] P. Crescenzi and V. Kann, “How to find the best approximation results – a follow-up to Garey and Johnson,” *ACM SIGACT News*, vol. 29, no. 4, pp. 90–97, 1998.
- [215] H. Mostaghimi Ghomi, “Three-dimensional knapsack problem with pre-placed boxes and vertical stability,” Ph.D. dissertation, University of Windsor, 2013.
- [216] P. Chu and J. Beasley, “A genetic algorithm for the multidimensional knapsack problem,” *Journal of Heuristics*, vol. 4, no. 1, pp. 63–86, Jun 1998.
- [217] U.S. NPSTC Public Safety Communications Report, “Public safety communications assessment 2012-2022,” The National Public Safety Telecommunications Council, Tech. Rep., 2012.
- [218] CEPT, “User requirements and spectrum needs for future European broadband PPDR systems (wide area networks),” ECC, Tech. Rep., 2013.
- [219] Critical Communications Broadband Group, “Mission critical mobile broadband : Practical standardisation & roadmap considerations,” TETRA and Critical Communications Association, Tech. Rep., 2013.
- [220] R. Ferrus, O. Sallent, G. Baldini, and L. Goratti, “LTE: The technology driver for future public safety communications,” *Communications Magazine, IEEE*, vol. 51, no. 10, pp. 154–161, October 2013.

- [221] R. Ferrus and O. Sallent, “Extending the LTE/LTE-A business case: Mission- and business-critical mobile broadband communications,” *Vehicular Technology Magazine, IEEE*, vol. 9, no. 3, pp. 47–55, Sept 2014.
- [222] Critical Communications Broadband Group, “PPDR need for dedicated communications and why they cannot rely on public systems for mission critical communications,” TETRA and Critical Communications Association, Tech. Rep., 2013.
- [223] ETSI, “Additional spectrum requirements for future public safety and security (PSS) wireless communication systems in the UHF frequency range,” ETSI, Tech. Rep. TR 102628, 2010.
- [224] Critical Communications Broadband Group, “Mobile broadband for critical communications users: A review of options for delivering mission critical solutions,” TETRA and Critical Communications Association, Tech. Rep., 2013.
- [225] Ericson, “Public safety mobile broadband,” Ericson, Tech. Rep. 305 23-3228, 2014.
- [226] Blue light mobile service. [Online]. Available: <http://bluelightmobile.be/en/home> [Last accessed: 09/28/2017]
- [227] ETSI, “ETSI TS 136 413,” ETSI, Tech. Rep., 2013.
- [228] 3rd Generation Partnership Project, “3GPP Technical Specification,” 3GPP, Tech. Rep., 2014.
- [229] Cisco, “Cisco visual networking index : Global mobile data traffic forecast update 2014-2019,” Cisco, Tech. Rep., 2014.

