TECHNISCHE UNIVERSITÄT
ILMENAU

Dissertation

Facilitating Flexible Link Layer Protocols for Future Wireless Communication Systems

---

Zur Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)

vorgelegt an der
Fakultät für Informatik und Automatisierung
Technische Universität Ilmenau

von
Dipl.-Inf. André Puschmann
geboren am 6. Dezember 1983 in Erfurt

Tag der Einreichung:	5. März 2015
Tag der Verteidigung:	29. Juli 2015
Gutachter:	Prof. Dr.-Ing. habil. Andreas Mitschele-Thiel Technische Universität Ilmenau
	Prof. Dr. rer. nat. habil. Jochen Seitz Technische Universität Ilmenau
	Prof. Luiz A. DaSilva, Ph.D. Trinity College Dublin

urn:nbn:de:gbv:ilm1-2015000329

TECHNISCHE UNIVERSITÄT
ILMENAU

Dissertation

Facilitating Flexible Link Layer Protocols for Future Wireless Communication Systems

---

to obtain the academic degree
Doktor-Ingenieur (Dr.-Ing.)

submitted to
Faculty of Computer Science and Automation
Ilmenau University of Technology

presented by
Dipl.-Inf. André Puschmann
born December 6, 1983 in Erfurt, Germany

Abstract

This dissertation addresses the problem of designing link layer protocols which are flexible enough to accommodate the demands of future wireless communication systems (FWCS). We show that entire link layer protocols with diverse requirements and responsibilities can be composed out of reconfigurable and reusable components. We demonstrate this by designing and implementing a novel concept termed *Flexible Link Layer* (FLL) architecture. Through extensive simulations and practical experiments, we evaluate a prototype of the suggested architecture in both fixed-spectrum and dynamic spectrum access (DSA) networks.

FWCS are expected to overcome diverse challenges including the continual growth in traffic volume and number of connected devices. Furthermore, they are envisioned to support a wide range of new application requirements and operating conditions. Technology trends, including smart homes, communicating machines, and vehicular networks, will not only grow on a scale that once was unimaginable, they will also become the predominant communication paradigm, eventually surpassing today's human-produced network traffic.

In order for this to become reality, today's systems have to evolve in many ways. They have to exploit allocated resources in a more efficient and energy-conscious manner. In addition to that, new methods for spectrum access and resource sharing need to be deployed. Having the diversification of applications and network conditions in mind, flexibility at all layers of a communication system is of paramount importance in order to meet the desired goals.

However, traditional communication systems are often designed with specific and distinct applications in mind. Therefore, system designers can tailor communication systems according to fixed requirements and operating conditions, often resulting in highly optimized but inflexible systems. Among the core problems of such design is the mix of data transfer and management aspects. Such a combination of concerns clearly hinders the reuse and extension of existing protocols.

To overcome this problem, the key idea explored in this dissertation is a component-based design to facilitate the development of more flexible and versatile link layer protocols. Specifically, the FLL architecture, suggested in this dissertation, employs a generic, reconfigurable data transfer protocol around which one or more complementary protocols, called link layer applications, are responsible for management-related aspects of the layer.

To demonstrate the feasibility of the proposed approach, we have designed and implemented a prototype of the FLL architecture on the basis of a reconfigurable software defined radio (SDR) testbed. Employing the SDR prototype as well as computer simulations, this dissertation describes various experiments used to examine a range of link layer protocols for both fixed-spectrum and DSA networks.

This dissertation firstly outlines the challenges faced by FWCS and describes DSA as a possible technology component for their construction. It then specifies the requirements for future DSA systems that provide the basis for our further considerations. We then review the background on link layer protocols, survey related work on the construction of flexible protocol frameworks, and compare a range of actual link layer protocols and algorithms. Based on the results of this analysis, we design, implement, and evaluate the FLL architecture and a selection of actual link layer protocols.

We believe the findings of this dissertation add substantively to the existing literature on link layer protocol design and are valuable for theoreticians and experimentalists alike.

Zusammenfassung

Gegenstand der vorliegenden Dissertation ist der Entwurf flexibler Link Layer Protokolle für zukünftige drahtlose Kommunikationssysteme (ZDKS). Dabei wird gezeigt, dass Link Layer Protokolle mit unterschiedlichsten Anforderungsprofilen und vielseitigen Aufgabenspektren vollständig aus rekonfigurier- und wiederverwendbaren Komponenten konstruiert werden können. Mittelpunkt der Arbeit stellt die neu entwickelte *Flexible Link Layer* (FLL) Architektur dar. Unter Zuhilfenahme einer prototypischen Implementierung, wird durch eine Vielzahl von realen Experimenten und Computersimulationen in unterschiedlichen Netzwerk-Szenarien die Funktionsfähigkeit sowie der praktische Nutzen unter Beweis gestellt.

ZDKS müssen eine Vielzahl technologischer Herausforderungen bewältigen. Hierzu zählen unter anderem das weiter steigende Datenaufkommen sowie die fortwährend steigende Anzahl von Kommunikationsendgeräten. Des weiteren wird von ZDKS erwartet, dass diese ein breites Spektrum an unterschiedlichen Anforderungen und Anwendungsgebieten unterstützen. Hierzu zählen beispielsweise intelligente Häuser, Fabriken und Autos. Die Benutzung dieser Technologien wird in naher Zukunft nicht nur in einem bislang unvorstellbaren Maße ansteigen, sondern das gesamte Kommunikationsverhalten revolutionieren. Dabei wird erwartet, dass die durch kommunizierende Maschinen generierte Datenmenge das von Menschen erzeugte Aufkommen übersteigen wird.

Damit diese Entwicklung vollzogen werden kann, müssen heutige Kommunikationssysteme vielseitig weiterentwickelt werden. Zum Einen müssen die Systeme die bereitgestellten Ressourcen effizienter und energiebewusster einsetzen. Zusätzlich dazu müssen sich neuartige Zugriffsarten für die gemeinsame Nutzung bzw. Mitbenutzung der spektralen Ressourcen etablieren. Dabei werden unter anderem sogenannte Dynamic Spectrum Access (DSA) Systeme eine Rolle spielen.

Um die genannten Ziele zu erreichen, ist die Flexibilität bzw. der flexible Einsatz aller Komponenten in ZDKS, unter Berücksichtigung der zuvor erwähnten Herausforderungen, von höchster Bedeutung. Herkömmliche Kommunikationssysteme werden jedoch üblicherweise mit spezifischen Anforderungen und Applikationsszenarien entworfen. Diese Anforderungen erlauben es ihren Entwicklern, speziell zugeschnittene Systeme zu konstruieren. Obwohl diese sehr effizient arbeiten, sind sie oftmals zweckgebunden und damit statisch und unflexibel. Ein Grund der fehlenden Flexibilität ist die Vermischung verschiedener Zuständigkeiten und Funktionen innerhalb eines Protokolls. Diese Vermischung erschwert die Erweiterung bestehender Protokolle sowie die Wiederverwendung von Protokollkomponenten deutlich.

Die in der vorliegenden Dissertation verfolgte Kernidee besteht daher in der Anwendung eines Komponenten-basierten Entwurfsansatzes, um die Entwicklung von flexiblen und vielseitig nutzbaren Link Layer Protokollen zu ermöglichen. Im Besonderen wird die Zweckmäßigkeit eines

generischen, rekonfigurierbaren Datentransferprotokolls untersucht. Dieses Protokoll, welches zentraler Bestandteil der vorgeschlagenen Architektur ist, wird dabei durch verschiedene Link Layer Applikationen komplementiert. Letztere übernehmen dabei managementorientierte Funktionen der Protokollschicht.

Um die Durchführbarkeit dieses Konzepts zu demonstrieren, wurde im Rahmen dieser Arbeit ein Prototyp der FLL-Architektur auf Software Defined Radio (SDR) Basis entwickelt. Unter Zuhilfenahme dieses Prototyps sowie durch den Einsatz von Computersimulation wurden im Kontext dieser Arbeit eine Reihe von Link Layer Protokollen entworfen, implementiert und evaluiert. Der Anwendungsbereich dieser Protokolle umfasst sowohl herkömmliche bzw. statische Netze, als auch DSA-Szenarien.

Hierzu werden zunächst die Herausforderungen von ZDKS sowie DSA als ein möglicher Technologiebaustein dieser beschrieben. Im Anschluss daran werden funktionale sowie nicht-funktionale Anforderungen an ZDKS aufgestellt.

Diese bilden die Grundlage für die Analyse und Bewertung bestehender Ansätze sowie den Entwurf der FLL-Architektur. Ausgehend von diesem abstrakten Architekturmodell wird im Anschluss ein Prototyp auf SDR Basis, sowie eine Auswahl an Link Layer Protokollen realisiert.

Die vorliegende Arbeit beschreibt und diskutiert somit einen neuartigen Ansatz zum Entwurf und zur Konstruktion von Link Layer Protokollen für ZDKS, welcher sowohl für die theoretische, als auch die experimentelle Forschung von Interesse ist.

Acknowledgments

First of all, I'd like to thank my supervisor Prof. Andreas Mitschele-Thiel for his support, for giving me the opportunity to pursue a Ph.D., and for providing me the freedom to explore my ideas.

I would also like to thank Prof. Jochen Seitz and Prof. Luiz A. DaSilva for agreeing to review my thesis and for being a member of the examination board.

I'd like to thank Carl Zeiss Stiftung and Deutsche Forschungsgemeinschaft (DFG) for the financial support of this research and for providing the opportunity to discuss problems, exchange ideas, and collaborate with scientists around the world. During the past four years, I was fortunate enough to get to know a great number of brilliant people at various conferences, workshops, and other events.

My gratitude is also due to the open-source community in general, all "SDR hackers", and various open-source projects that were a prerequisite for this research. In particular, I'd like to thank the developers and contributors of *Iris*, *liquid-dsp*, and *GNU Radio*. I'd also like to thank Paolo Di Francesco and Bastian Blößl for fruitful discussions, valuable comments, and their kind cooperation.

My gratitude is due to all members of the Integrated Communication Systems (ICS) group and the International Graduate School on Mobile Communication (Mobicom). To Mohamed A. Kalil, for his guidance in the beginning, for being great company in the office, and for helping to kick-start the "paper machine". To Florian Liers, for many fruitful discussions and for teaching me to become a true critic. To Michael Grimm, for helping to catch up with electrical engineering and RF bits that puzzled me. To Tobias Simon, for all the inspiration and fun things we did during our time at and besides university, and for also being addicted to coffee. Any idea how many we had over the last ten years?

Of course, this can't be complete without mentioning the lovely people outside work: Basti, Dave, John, Hendrik, Marcus, and their partners and families. It's great and always special to me to spend time and enjoy life with you guys.

My deepest gratitude is due to my family: my brother Thomas for always being the one I can count on, and my parents Jürgen and Kornelia for their unwavering support and for teaching me the truly important things in life. This thesis is dedicated to both of you.

Finally, I am deeply thankful to Janette, for all the love and support, and for giving me the strength and motivation to remain persistent. I couldn't have finished this without you.

Contents

1 Introduction

Personal mobile wireless communication has undergone a dramatic development since its introduction in the late seventies and early eighties of the last century. What began with an analog cellular system for voice communication, has evolved over several generations. Wireless communication systems are now delivering a multitude of services and facilitate unconstrained cooperation and information access anywhere in the world [1]. In the future, this trend is expected to continue such that, by 2018, there will be more traffic originating from wireless and mobile devices than from wired devices [2]. Not only will we see continual growth in traffic volume, but also a massive growth in the number of connected devices. This development is explained by faster network connectivity as well as by cheaper and more powerful devices. This success opens up new application domains that may have highly diverse requirements, largely differing among each other as well as from today's human-centric communication paradigm [3]. This evolution towards communicating machines, often termed the Internet of Things (IoT), as well as the continuing blurring between fixed and mobile wireless networks will be tremendously exciting, but will also pose significant challenges on future wireless communication systems (FWCS). It is believed that the desired goals can only be achieved through the continuous evolution of existing systems in combination with the introduction of revolutionary new concepts. These concepts include new methods for accessing and sharing available resources as well as novel communication paradigms, such as device-to-device communication. But how, exactly, can future systems be designed?

1.1 Motivation

Traditional communication systems have been developed with concrete and distinct applications in mind. Hence, system designers and engineers were able to tailor them according to fixed requirements and operating conditions, often resulting in highly optimized but inflexible systems. However, as FWCS are expected to support a much wider range of applications and to operate under much more diverse network conditions, flexibility is of paramount importance. In recent years, the design of flexible and reconfigurable physical (PHY) layer techniques has played a significant role in both academia and industry. This includes the development of wideband transceivers and antennas, as well as advanced communication techniques, reconfigurable waveforms, and software defined radios (SDRs).

However, as the number of modifiable parameters of a communication system grows, management-related aspects become increasingly important. This is primarily because the particular values of those parameters need to be agreed upon for successful operation, before and during communication. Being the system entity on top of a flexible PHY layer, the link layer of a communication system naturally becomes part of the challenge and any potential solution. Thus, the central research question asked by this dissertation is whether it is possible to develop a generic link layer architecture that accommodates the flexibility needs of FWCS?

Towards answering this question, we have developed the *Flexible Link Layer* (FLL) architecture: a component-based design that partitions its functionality into multiple, reusable building blocks. Specifically, we propose to deploy a generic and reconfigurable data transfer protocol at the core of a link layer configuration. Multiple complementary protocols, called link layer applications, are placed around the core protocol to cater for management-related aspects of the layer. This novel design allows for reuse and recombination of existing components to entirely new and potentially unforeseen protocols without reimplementing common functionality from scratch. In particular, this dissertation focuses on a new class of communication systems to which the proposed concept is applied. This family of communication systems, called dynamic spectrum access (DSA) systems, introduces new challenges of its own. It is shown how those challenges are addressed by the architecture, without neglecting the evolution of existing fixed-spectrum networks.

Besides the conceptual framework, this dissertation explicitly focuses on practical realization and validation. When appropriate, we employ computer simulations for large-scale system evaluation. Furthermore, we conduct real-world experiments in a SDR testbed. Even if small-scale, we believe that this kind of proof-of-concept implementation provides valuable insight into the practical realization of FWCS and, therefore, forms an indispensable tool for their development. To the best of our knowledge, there is no prior work on the design, implementation, and practical validation of a flexible link layer architecture that facilitates the actual realization of both adaptive fixed-spectrum protocols and DSA communication systems.

1.2 Contributions

In addressing the issue of designing flexible link layer protocols for FWCS, the contributions of this thesis span several aspects:

- **Analysis and Definition of Link Layer Requirements**

  The challenges that will be faced by wireless communication systems in order to address the expectations of the future are outlined. In the area of system and protocol design, the effects of diversified requirements are highlighted. It is shown that, as communication systems will see applications with a much wider range of requirements and characteristics, the need for flexibility, reusability, and complexity reduction during the entire life-cycle of the communication system is also increased. With a focus on DSA, the issue of initially establishing and maintaining a link for seamless communication is emphasized. This development motivates the design of a new protocol architecture that is both capable of supporting a wider range of requirements and flexible enough to be tailored to the needs of a given application.

- **Analysis, Functional Decomposition/Recombination, and Architectural Design**

  Current protocols and protocol frameworks proposed by the research community are analyzed with respect to their commonalities and differences, architectural design, and placement of functions therein. The main disadvantages of monolithic protocols are described. They are then decomposed into their fundamental elements and a novel concept for recombining them is suggested. This concept, termed the FLL architecture, employs a component-based design for building flexible and reusable link layer protocols while keeping their complexity manageable.

- **Prototype Implementation**

  A substantial part of this dissertation consists of practical experiments that are used to validate, evaluate, and compare its findings. Using a reconfigurable software radio architecture, the design and the implementation of an FLL prototype including more than half a dozen link layer protocol components for various purposes are described. This includes data transfer, medium access control, link establishment, and spectrum mobility protocols.

- **Practical Evaluation and Performance Analysis**

  Using this prototype implementation as well as computer simulations, the dissertation describes a range of experiments used to both showcase the feasibility of the proposed approach and to examine the performance of the prototype and the entire link layer ecosystem. The results of the experiments are discussed and used to emphasize the benefits of flexible, component-based link layer protocols for building FWCS.
  As one example of such a flexible link layer, the development of a load-adaptive protocol that supports switching between different MAC schemes in response to network condition changes is shown. In the considered scenario, this load-adaptive protocol achieves up to 32% higher average throughput than the non-adaptive solution.
  This dissertation also makes a number of contributions in the field of MAC protocol implementation using SDRs. For example, it is demonstrated that CSMA-MACs implemented using SDR technology and commodity IEEE 802.11 hardware exhibit a similar behavior, even if the SDR protocol and signal processing is entirely implemented in software.

1.3 Restriction of Scope

The research topic of this thesis, in general, spans a wide range of areas of electrical engineering and computer science. These areas include topics from signal processing, communication protocol design, and software architectures. The actual scope of the thesis, however, is limited to link layer issues of overlay DSA systems in the context of experimental SDRs. As such, the research doesn't enhance PHY layer signal processing techniques, routing or transport protocols. It also doesn't cover security or advances quality of service related aspects.

Even though building upon the idea of a component-based design, the components themselves are not the core focus of this dissertation. Instead of developing one specific protocol, the FLL architecture aims to provide a generic framework that may be used for a great number of possibly unforeseen use cases in link layer research. Although we design, implement, and evaluate a number of novel protocols that exploit the capabilities of the architecture, their underlying algorithms are not a direct contribution of this thesis.

1.4 Thesis Structure

The thesis is organized as follows: Chapter 2 provides the reader the necessary background knowledge of the dissertation. The first part outlines the challenges that will be faced by wireless communication systems in order to address the expectations of the future. To overcome them, the chapter presents cognitive radio (CR) and DSA as possible technology components to build future systems. The importance of flexibility is emphasized in order to handle the wide range of requirements and characteristics of future applications.

Chapter 3 discusses related work in the research field. The first part of the chapter surveys flexible protocol frameworks and architectures, i.e., work that deals with the design and development of protocols in general. The second part presents a selection of specific protocols that are of interest to this thesis. They are discussed with respect to functional as well as architectural aspects. Furthermore, their main shortcomings are outlined.

Following that, Chapter 4 describes the FLL architecture, our approach for addressing some of the existing issues. We describe the design and the main components of our architecture and also provide an initial qualitative analysis.

Chapter 5 evaluates the proposed FLL architecture in greater detail. First, the implementation of a prototype using a reconfigurable SDR framework is examined. This includes the architecture itself as well as a number of general purpose link layer protocols. Furthermore, its applicability for developing adaptive protocols for both dynamic and fixed spectrum access systems is demonstrated through various experiments.

Finally, Chapter 6 summarizes our findings and suggests areas for future investigation.

1.5 Publications Record

The following publications relate directly to this dissertation:

- André Puschmann and Andreas Mitschele-Thiel. Modeling Management Functions as Link Layer Applications. In *Networked Systems (NetSys)*, pages 1–2, Cottbus, Germany, March 2015

- André Puschmann and Andreas Mitschele-Thiel. Implementation and Evaluation of a Flexible, Load-Adaptive Link Layer Protocol. In *20th Annual International Conference on Mobile Computing and Networking (MobiCom), 9th ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH)*, pages 73–80, Maui, HI, USA, September 2014

- André Puschmann, Paolo Di Francesco, Mohamed A. Kalil, Luiz A. DaSilva, and Andreas Mitschele-Thiel. Enhancing the Performance of Random Access MAC Protocols for Low-cost SDRs. In *19th Annual International Conference on Mobile Computing and Networking (MobiCom), 8th International Workshop on Wireless Network Testbeds Experimental Evaluation and Characterization (WiNTECH)*, pages 9–16, Miami, FL, USA, September 2013

- André Puschmann, Mohamed A. Kalil, and Andreas Mitschele-Thiel. A Component-based Approach for Constructing Flexible Link-Layer Protocols. In *8th International Conference*

on Cognitive Radio Oriented Wireless Networks (CROWNCOM), pages 244–249, Washington D.C., USA, July 2013

- André Puschmann, Shah N. Khan, Mohamed A. Kalil, and Andreas Mitschele-Thiel. Database-assisted Coordinator-based Spectrum Mobility in Cognitive Radio Ad-hoc Networks. In *10th International Symposium on Wireless Communication Systems (ISWCS), Ilmenau, Germany, August 2013*, pages 1–2, Ilmenau, Germany, August 2013

- André Puschmann, Mohamed A. Kalil, and Andreas Mitschele-Thiel. A Flexible CSMA based MAC Protocol for Software Defined Radios. *Frequenz Journal of RF-Engineering and Telecommunications*, 6:261–268, October 2012

- André Puschmann, Mohamed A. Kalil, and Andreas Mitschele-Thiel. Implementation and Evaluation of a Practical SDR Testbed. In *4th International Conference on Cognitive Radio and Advanced Spectrum Management (COGART)*, pages 1–5, Barcelona, Spain, October 2011

The following publications are the result of collaborative work related to the content of this dissertation:

- Bastian Bloessl, André Puschmann, Christoph Sommer, and Falko Dressler. Timings Matter: Standard Compliant IEEE 802.11 Channel Access for a Fully Software-based SDR Architecture. In *20th Annual International Conference on Mobile Computing and Networking (MobiCom), 9th ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH)*, pages 57–64, Maui, HI, USA, September 2014

- Tobias Simon, André Puschmann, Shah N. Khan, and Andreas Mitschele-Thiel. A Lightweight Message-based Inter-Component Communication Infrastructure. In *5th International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN2013)*, pages 145–152, Madrid, Spain, June 2013

- André Puschmann and Mohamed A. Kalil. The Impact of a Dedicated Sensing Engine on a SDR Implementation of the CSMA Protocol. In *10th International Symposium on Wireless Communication Systems (ISWCS)*, pages 1–5, Ilmenau, Germany, August 2013

- André Puschmann, Shah N. Khan, Ali H. Mahdi, Mohamed A. Kalil, and Andreas Mitschele-Thiel. An Architecture for Cognitive Radio Ad-Hoc Network Nodes. In *12th International Symposium on Communications and Information Technologies (ISCIT)*, pages 1–5, Gold Coast, Australia, October 2012

- Mohamed A. Kalil, André Puschmann, and Andreas Mitschele-Thiel. SWITCH: A Multichannel MAC Protocol for Cognitive Radio Ad Hoc Network. In *IEEE 76th Vehicular Technology Conference (VTC)*, pages 1–5, Québec City, Canada, September 2012

2 Background

This chapter first looks at FWCS and outlines the challenges that they will face. It then presents technology components that may be used to build them. CR, DSA, and SDR are then introduced to provide a basis for the further understanding of the thesis. Before discussing link layer aspects of future systems, we discuss and analyze the evolution of link layer protocols in traditional communication systems.

2.1 Future Wireless Communication Systems

According to Cisco's Visual Networking Index [2], the year 2018 will mark a turning point for wireless communication. For the first time, traffic from wireless and mobile devices will exceed the traffic from wired devices. Furthermore, over half of all IP traffic in the Internet will originate from non-PC devices, such as tablets, smartphones, and machine-to-machine (M2M) modules. While PC originated traffic will increase by only 10%, M2M type communication will have growth rates of 84%. This trend can be explained by cheaper, more powerful devices and faster network connectivity that are driving the transition of our society into a fully connected, always connected society. We will see a rapid growth in the number of connected devices and new application fields such as smart homes, smart factories, smart cars, and smart infrastructure. This evolution towards communicating machines, often termed the Internet of Things (IoT), as well as the continuing blurring between mobile and fixed wireless networks will be tremendously exciting, but will also pose significant challenges on FWCS.

2.1.1 Challenges

According to Baldemair *et al.* [3], the key challenges that need to be addressed by FWCS can be summarized as follows:

- **Massive Growth in Traffic Volume**

  The amount of traffic that will be carried over our communication networks will undoubtedly continue to grow in the future. Cisco's forecast expects that the global IP traffic will increase threefold over the next five years and will surpass the zettabyte threshold in 2016.

In other words, in the year 2016, over 91.3 million terabytes of IP traffic will be transferred every month [2]. As has already been outlined above, part of this growth will be driven by M2M-type communication.

- **Massive Growth in the Number of Connected Devices**

  The same type of applications, i.e., communicating machines, will also cause a tremendous increase in the number of connected devices. This alone poses significant challenges on FWCS.

- **Wide Range of Requirements and Characteristics**

  The third challenge is the wide range of requirements and characteristics that FWCS will have to support. FWCS have to provide solutions for the increasing demand for higher data rate and ubiquitous connectivity. Furthermore, the advance of M2M type communication brings up new applications and use cases that largely differ among each other as well as from today's human-centric communication paradigm.

  Consider the example of communication delay, also provided by Baldemair *et al.* [3]. Some application areas, such as vehicular networks or industrial automation, may have very strict temporal requirements due to their safety-critical character. Contrary to that, other IoT applications, such as electricity meters in smart homes, may have loose requirements when it comes to communication delay. FWCS, however, are intended to support both use cases.

- **Costs and Sustainability**

  Despite the expected growth, costs for both network operators and end users will play a major role for FWCS. For operators, costs for deploying, operating, and maintaining the infrastructure are of primary importance. New technologies need to be designed with energy consumption and resource efficiency in mind [16].

2.1.2 Technology Components

Today's wireless networks have to evolve in many ways to meet the expectations of the future. This development is believed to take place in two, not necessarily sequential, phases [3]. At first, we will see a continued evolution and further optimization of existing technologies, such as Long-Term Evolution (LTE) or wireless local area networks (WLANs). Second, we will experience the introduction of new wireless access technologies, maybe because the required changes would be too severe for current systems.

The key technology components that, either alone or combined, provide the necessary potential for the transition into the connected society, may be summarized as follows [3]:

- **New Spectrum Access Methods and Frequencies**

  The scarcity and inefficient assignment of radio spectrum is one of the biggest problems in today's wireless networks. In the future, we are very likely to see new methods for accessing radio spectrum. One possible approach is to allow spectrum to be shared among radio users. Similar to a car or bicycle sharing system in which a shared resource, the radio spectrum in our case, is made available for use to individuals on a short term basis.

Furthermore, the future will bring up radios operating in much higher frequency bands than today's systems. The use of spectrum in the 30 GHz band and beyond would provide a massive amount of transmission bandwidth that is required to satisfy the traffic volume growth.

- **Massive Antenna Configurations and Ultradense Deployments**

  The explosive growth in high data rate applications also requires space to be used more efficiently. One possible approach for providing higher system capacity and for reducing interference is the use of configurable antennas. This would allow the use of advanced technologies, such as spatial multiplexing as well as sender and receiver beamforming.

  Another possibility to tackle the high data rate and low energy consumption goals is network densification. This is especially useful in situations in which a large number of users with high traffic needs is present in a small area, such as in offices, public places, and shopping malls.

- **Device-to-Device Communication**

  Infrastructure-based architectures dominate today's communication networks. Cellular systems, such as Global System for Mobile Communications (GSM), LTE, and WLANs are prominent examples of this kind of network architecture.

  However, in the future, we will also see infrastructure-based networks that support device-to-device communication (D2D) as well as relaying between devices with a benefit for both network operators and end users.

  Use cases for direct communication between devices are manifold. However, it appears particularly advantageous in scenarios in which information is produced and consumed only locally. Possible examples include M2M communication in factories, vehicular networks, and public safety networks.

2.1.3 An Initial Concern: The Need for Flexibility

The dream of an always connected society introduces exciting opportunities for new applications, new business models, and stakeholders; but it also holds significant challenges for researchers and engineers to deal with. We believe that the diversification of applications and operating environments is particularly challenging for FWCS.

Traditionally, communication systems are designed for specific applications with a fixed set of functional requirements. This allows system designers and engineers to tailor them accordingly. However, FWCS are intended to be more general purpose in the sense that they support a wider range of applications. At the same time, they are expected to provide better performance than their predecessors in all possible circumstances, i.e., they are expected to be more specialized. From a system engineer's point of view, a conflict between goals arises because one of them calls for specialization whereas the other calls for generalization. However, both cannot be ignored.

We therefore argue that system flexibility, i.e., the system's ability to respond to internal or external changes by adapting its own structure and behavior, is of paramount importance in meeting the desired goals [17, 3].

Examples of events that may cause a communication system to adapt its behavior include changes to the application requirements (internal) or to the operation environment (external). Ideally, the flexibility should span over the entire architecture of the communication system and should be available throughout the entire life-cycle of a device. This includes the PHY layer of a communication system as well as higher protocol layers, such as the link and network layer.

This dissertation addresses flexibility at the link layer of FWCS. Specifically, it focuses on how the required flexibility for adapting existing link layer protocols and for supporting new spectrum access methods can be achieved. In the next section, we will briefly explain how spectrum access is organized nowadays and why this may be problematic in the future. We will then present CR and DSA systems that have been proposed as possible approaches for addressing the challenges mentioned above.

2.2 Future Spectrum Access Paradigms and Technologies

The fixed assignment of frequency resources has served well for many years. However, the tremendous growth in use of wireless technologies across the world and the resulting resource scarcity have unveiled the substantial disadvantages of this approach. The biggest drawback stems from the fact that frequency bands are usually assigned to license holders on a long term basis for large geographical regions. The high temporal and spatial variability results in large swaths of spectrum that are only occupied sporadically, whereas other parts of the spectrum experience heavy usage.

For example, the 900 or 1800 MHz bands for cellular communication and the 2.4 GHz Industrial, Scientific and Medical (ISM) band are heavily overloaded. Furthermore, spectrum occupancy also varies with measurement location (indoor or outdoor), geographic characteristic, and population density (urban, suburban, or rural) [18]. According to the U.S. Federal Communications Commission (FCC), these factors lead to an overall variation of the spectrum utilization between 15% and 85% [19]. Similar observations have also been made from measurement campaigns carried in Ilmenau, Germany. For example, Figure 2.1 illustrates the power spectral density of the wireless spectrum between 70 MHz to 1010 MHz. From the obtained results, it can be observed that the cellular spectrum between 900 Mhz and 1 GHz is heavily used, while the spectrum below, i.e., between 750-900 MHz, is only lightly used.

The limited availability of spectral resources and their inefficient usage in today's communication networks requires a new paradigm in order to better exploit and utilize the available communication resources in tomorrow's wireless networks.

2.2.1 Cognitive Radio

A possible solution to the aforementioned issues is the concept of a Cognitive Radio (CR). CR is a technology that promises to allow a better utilization of the available spectrum by dynamically adapting its operating parameters based on changes in its operation environment. Several definitions of CR have evolved since Mitola coined the term in his Ph.D thesis [20]. A report from the *SDR Forum* [21] alone states more than nine different interpretations, including one offered by Haykin [22], which best fits how this thesis views CR:

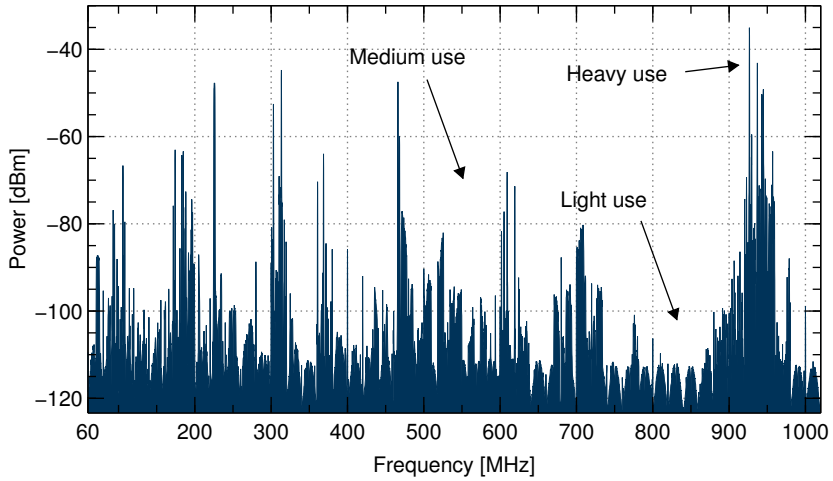


Figure 2.1: *Average power spectral density plot of a measurement campaign between 70 MHz to 1010 MHz carried out by Michael Grimm on June 29, 2010 in Ilmenau, Germany using a wide band horn antenna.*

> *Cognitive radio is an intelligent wireless communication system that is aware of its surrounding environment (i.e., outside world), and uses the methodology of understanding-by-building to learn from the environment and adapt its internal states to statistical variations in the incoming RF stimuli by making corresponding changes in certain operating parameters (e.g., transmit-power, carrier-frequency, and modulation strategy) in real-time, with two primary objectives in mind:*
>
> - *highly reliable communications whenever and wherever needed;*
> - *efficient utilization of the radio spectrum.*

From Haykin's definition, three important aspects can be deduced that are inseparably connected to a CR:

- learning and self-adaptation,
- environmental awareness, and
- resource efficiency.

2.2.2 Dynamic Spectrum Access

One possible and certainly today's most popular use case of CR technology is dynamic spectrum access (DSA). To understand the idea behind DSA, consider the following example. Given the inefficient and unbalanced allocation of frequency resources, wouldn't it be possible to relieve heavily occupied frequency bands at times by offloading parts of the traffic to other, less occupied bands? This is exactly the idea behind DSA. Thereby, the CR is viewed as a so-called unlicensed user or secondary user (SU) that borrows, rents, or leases spectrum from a spectrum licensee, also called licensed user or primary user (PU) [23]. In this form of licensed DSA, the PU permits the SU

to make use of the spectrum at times when the PU is not using it. It also implies that the spectrum should be returned to the PU, if the PU attempts to use it. This behavior is referred to as spectrum mobility because the spectrum used for communication changes during operation. Although spectrum mobility is strongly associated with DSA, it is also applicable in non-DSA scenarios. For example, due to changes in the radio environment causing unacceptable variations in quality of service (QoS). Interestingly, there appears to be quite some confusion about the distinction of CR and DSA among researchers [24]. This confusion led to a considerable number of publications on CR that are, in fact, DSA only. O'Sullivan put it aptly when he wrote that DSA is commonly "considered to be the application and a CR the radio employed as part of it" [24, p. 7].

Figure 2.2 illustrates a simple representation of an *overlay* DSA scenario in which frequency resources that are not occupied by PUs are opportunistically used by a spectrum agile SU. In this figure, it can also be observed that at some point in time, multiple spectral opportunities (or white spaces) may be available whereas at other times, there may be no opportunities at all.

It should be noted that there also exists another form of DSA called *underlay* DSA. Using an underlay sharing scheme, the SU accesses the same frequency band at the same time as the PU. However, instead of using a high transmit power, the SU is only allowed to transmit with low power levels such that it does not interfere with the PU [25]. Underlay DSA usually requires some kind of cooperation between PUs and SUs in order to define and control the allowed degree of interference [19].

The classification scheme described above only differentiates between two paradigms, i.e., underlay and overlay. However, it should be noted that there is an alternate scheme that differentiates between three different access paradigms: underlay, overlay, and interweave [26]. The underlay paradigm is identical in both schemes, mandating that the interference created at the PU receiver should remain below some predefined threshold. Furthermore, the interweave paradigm of the alternate scheme is identical to the overlay paradigm of the basic scheme, whereby SUs and PUs only occupy resources that are orthogonal to each other. For the overlay paradigm, however, the alternate scheme assumes that the SU possesses a certain degree of information about the PU that may be used to limit the interference caused to it, or even to assist it by relying its transmissions. This information may include knowledge about the PU channel gains, the employed encoding techniques, and possibly even the transmitted data sequences [26].

According to Akyildiz *et al.* [19], the main functions of a DSA-CR can be summarized as follows:

- **Spectrum sensing:** Spectrum sensing is the process of detecting unused or partially unused spectrum bands.
- **Spectrum management:** Spectrum management is the process of regulating the use of the spectrum to best meet the user communication requirements.
- **Spectrum mobility:** Spectrum mobility is the process by which a CR changes its spectrum used for communication during operation.
- **Spectrum sharing:** Spectrum sharing provides a means to share the spectrum among all coexisting radio users.

This dissertation views a CR, in its last development stage as envisioned by the above definition, as an idealistic radio that may be difficult to realize in the near future. We believe it is more likely that some of the concepts and algorithms developed under the umbrella of CR become part

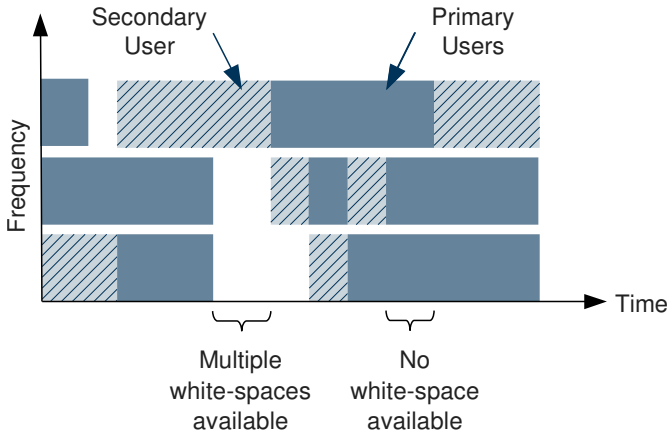


Figure 2.2: *Sketch of an overlay DSA scenario in which frequency resources not used by PUs (light blue) may be exploited by the SU.*

of FWCS, even if not explicitly termed CR. Fully or partially, concepts for environmental awareness, learning, self-adaptation, and resource-efficiency will be integrated in successive system generations. In fact, evidence for this to happen in the near future can already be observed from the recent discussion on accessing unlicensed spectrum in upcoming releases of LTE [27].

This dissertation takes up the spirit of a DSA-CR as one technology component and explores the role of the radio's link layer in achieving the desired goals for FWCS. However, in order for them to become reality, the entire radio architecture has to be configurable and adaptable. In the next section, we will, therefore, explore how adaptive radios may be designed from a PHY layer perspective.

2.2.3 Software Defined Radio

Traditional communication systems are designed with concrete and distinct applications in mind. This allows system designers and engineers to tailor them according to fixed requirements and operating conditions. To build them, highly optimized off-the-shelf and application-specific components are used in order to optimize their cost, energy-efficiency, and performance. Once the design has been finished and the radio has been fabricated, it is virtually impossible to modify any of its functions.

However, in order to realize the concepts envisioned by FWCS, it is of paramount importance to have an underlying communication system that provides the required amount of flexibility and reconfigurability. A SDR is such a device. According to the *SDR Forum* [21], a SDR is a "radio in which some or all of the physical layer functions are software defined".

The reconfigurability of SDRs can be achieved by implementing radio components by means of reprogrammable software, instead of using inflexible hardware components. This includes radio components such as mixers, filters, amplifiers, modulators, and demodulators. In a SDR, these components are implemented inside one or more field-programmable gate arrays (FPGAs), one or more digital signal processors (DSPs), one or more general-purpose processors (GPPs), or a combination thereof. Therefore, the operating characteristics of the radio, such as its frequency

band, modulation scheme, or coding rate, can be modified at run time [23, 28].

The term *software radio* has also been coined by Mitola [29]. Mitola draws the picture of an idealized software radio that, literally, consists of only an antenna, an analog-to-digital converter (ADC), and a digital-to-analog converter (DAC). In such a radio, all of the signal processing tasks are performed in the digital domain without any frequency conversion. Figure 2.3 illustrates the block diagram of an ideal software radio receiver. However, such a design still has a number of technical challenges that make it currently infeasible to carry out the AD/DA conversion at the antenna. These challenges include the high input bandwidth, high sampling rate, and high dynamic range.

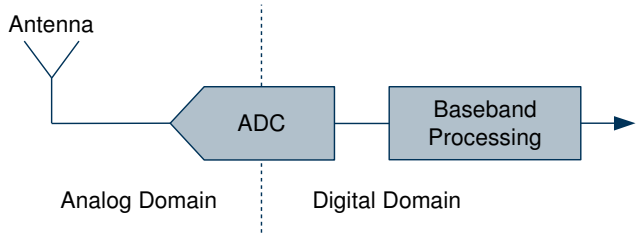


Figure 2.3: *Schematic block diagram of an ideal software radio receiver (reproduced from [29]).*

As the ideal software radio is not and will not be implementable in the near future, a more realistic approach is needed to achieve the desired flexibility nonetheless. One possible strategy is lowering the ADC/DAC requirements by keeping a minimum amount of analog components before the actual conversion stage. Possible analog components include radio frequency (RF) and intermediate frequency (IF) filter, frequency transposition, and several amplification stages, i.e., low-noise amplifier (LNA), power amplifier (PA), and automatic gain control (AGC). As with the ideal solution, the demodulation and decoding is performed by the digital part. This is a more practical design which is used by many SDRs nowadays [30].

Figure 2.4 illustrates the block diagram of a typical SDR transceiver architecture. In practice, probably the most widely used devices for SDR prototyping, in both industry and academia, is the Universal Software Radio Peripheral (USRP) [31].

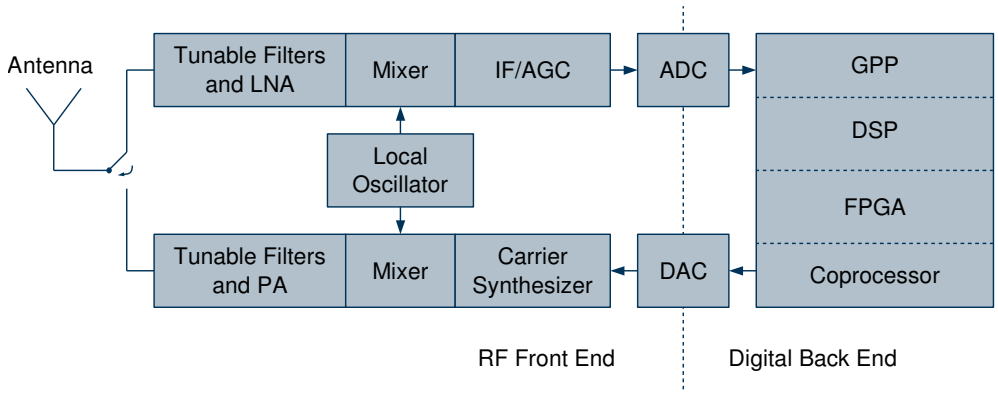


Figure 2.4: *Block diagram of a practical SDR hardware architecture (reproduced from [30]).*

Naturally, most of the research work on SDRs is PHY layer related. In the recent past, however,

research on higher protocol layers, employing a SDR as the underlying communication hardware, has gained a lot of attraction [32, 33, 34, 35]. Much of this work is concerned with delay and timing issues that arise through the separation of the RF front end and the digital back end. This is because many SDR architectures typically consist of two hardware components: a conventional host computer that implements the majority of the DSP functions and an external device that includes the ADC/DAC and the entire analog part of the radio.

In using SDR technology for experimentation and practical validation, this dissertation bridges the gap between conceptional work on link layer protocol architectures on one side, and research on advanced PHY layer concepts on the other side. But what, exactly, is a link layer protocol and what functions does it provide? The next section provides an answer to this question by analyzing the responsibilities of the link layer in communication systems.

2.3 Analysis and Evolution of Link Layer Protocols

Despite the different views on layering as seen by the IP suite and the open systems interconnection (OSI) model [36], the link layer is generally described to be the layer on top of the PHY layer and below the network layer (OSI model) or Internet layer (IP suite). In the broadest terms, the responsibility of the link layer (or data link layer) is to provide data communication services over a physical communication channel (lower layer) to the network layer (its users) [36]. In order to achieve this goal, it has to cater for three basic functions. While this may seem straightforward to carry out, this section shows that there are quite some challenges associated with it, not only because the aforementioned communication channel may be imperfect. In fact, we will discover that technological advances and innovations in communication networks have also introduced new link layer responsibilities. We will start this brief introduction by examining the basic functions.

2.3.1 Basic Functions

The basic functions of the link layer comprise three aspects that are strongly associated with the characteristics of the underlying PHY layer. They may be summarized as follows [36]:

- **Delimiting or framing:** One of the core functions of the link layer is to delimit the raw stream of information, i.e., to indicate its beginning and end. This is because the PHY layer operates on the granularity of bits and usually only takes those as input or provides them as output. In other words, the link layer provides a translation of the raw bitstream into logical data frames[1].

- **Error control:** In practice, communication over a physical medium is subject to noise. Therefore, errors may be introduced during communication between sender and receiver. Hence, dealing with transmission errors is a further core function of the link layer. This problem is usually dealt with by using an acknowledgment mechanism that tells the sender whether a frame has received its destination or not.

[1] Therefore, delimiting is sometimes also called framing.

- **Flow control:** Flow control is required to avoid overrunning a receiver with too many frames. This might happen if a fast transmitter is sending to a slow or overloaded receiver. If flow control is not present, an overloaded receiver would soon need to start dropping frames simply because it cannot handle them. This problem can be avoided by introducing some kind of feedback mechanism that an overloaded receiver uses to tell the transmitter to stop sending further frames.

2.3.2 From Point-To-Point to Broadcast Networks

During the early days of networking, nodes were usually connected through direct point-to-point links. Therefore, messages that were transmitted from one computer could be received by the corresponding node at the end of the line.

However, with the introduction of broadcast channels, this assumption was no longer true because a single transmitter can be heard my multiple receivers. This also creates opportunities for collisions that occur if multiple senders are transmitting at the same time. Therefore, in any broadcast network, one of the key challenges is to decide who gets to use the shared medium when there are multiple contending users for it. The mechanism used to solve this issue is called medium access control (MAC).

According to the seven-layer OSI model, MAC is part of the medium access sub-layer of the link layer. Because many of today's networks use broadcast channels, the boundaries between both sub-layers are blurred and, thus, the term MAC is often used to denote the entire layer, including the basic function discussed above. This includes LANs, WLANs, and cellular networks such as Global System for Mobile Communications (GSM) or Universal Mobile Telecommunications System (UMTS). It should be noted, however, that the functional separation in cellular networks like GSM or UMTS is usually much clearer than in LANs or WLANs. This problem is analyzed in more detail in Chapter 4 and 5 of this thesis.

As it has already been discussed above, the main goal of MAC is scheduling access to a shared medium among multiple users. The only possibility to achieve this goal is by "dividing" the communication resource such that multiple users can make use of it. In wired networks, this division can either happen in time or frequency[2]. In time-division multiple access (TDMA) strategies, time is divided into slots that are assigned to the individual nodes. This allows multiple stations to share a common medium. If frequency-division multiple access (FDMA) strategies are used, each user is given an individual part of the overall frequency spectrum such that multiple users can make use of it at the same time, by accessing their share.

2.3.3 From Wired to Wireless Networks

With the introduction of wireless networks, even more strategies for sharing access emerged. This includes the use of code-division multiple access (CDMA) schemes in which each transmitter is assigned a code with which the data signal is combined for the actual transmission.

A further access method that gained a lot of attention recently [37] is space-division multiple access (SDMA). Instead of simply radiating the signal in all directions, SDMA aims at better utilizing the available resources by creating directed transmission and radiation patterns. This

[2]Although using orthogonal codes would be possible, it is not widely used in wired networks.

allows, for example, to place multiple mobile nodes around a base station, each served with a highly-directed wireless communication link.

From the MAC point of view, further attention has been devoted to improve the performance of wireless networks. This includes strategies to avoid difficulties due to the hidden or exposed node problem as well as the optimization of timing characteristics.

2.3.4 From Static to Mobile Networks

Naturally, with the availability of wireless networks, it was also possible to carry around devices and to communicate on the move. Network nodes were no longer physically static but could change their geographic position. Furthermore, they could be connected to multiple networks, depending on their current location. This usually means that they also need different addresses because they are likely to be part of another administrative domain. Unfortunately, mobility is a feature with no well-defined place in the classical protocol stack [38]. This resulted in different approaches for implementing it in the link layer [39], network layer (e.g., Mobile IP [40]), transport layer [41], or variations and combinations of them.

Moreover, it is not uncommon in wireless networks to be connected to multiple networks at once[3]. For example, this might happen in cellular networks when users move along the edge of a network such that they are in the coverage area of two or more base stations at the same time. It might also happen in enterprise WLAN deployments with a large number of access points (APs), such as in universities, companies, or hotel buildings. If a node has multiple (connected) interfaces, it is said to be multi-homed. In fact, mobility can be viewed as a special case of multi-homing, called dynamic multi-homing, where a node disconnects from one network and connects to another while moving [42].

Even though it is difficult to argue that mobility is a link layer issue exclusively, there is some indication that both are highly related, as evidenced by the ongoing scientific research in that field [38, 42, 43]. However, if this is really the case, it is also clear that it needs to be managed and controlled somehow at the link layer. But wouldn't that also mean that supporting mobility would be a new link layer function?

2.3.5 Large Scale Mobile and Constraint Wireless Networks

With a world wide trend for decreasing the overall energy consumption, energy efficiency in computing in general, and in wireless networking in particular, gained tremendous interest. In the area of link layer protocols, this caused the development of a great number of new energy-efficient protocols or variations of existing proposals. Especially in the domain of wireless sensor networks (WSN), reducing energy consumption is of particular importance because nodes in a WSN are usually battery-powered. With energy becoming a scarce resource, researchers came up with techniques that would turn off the network interface at certain times in order to save energy. Imagine a case where two nodes wish to communicate with each other and both alternating their on/off-periods. In such a situation, communication would be impossible because whenever node A is awake trying to send a message to node B, node B cannot receive anything, simply because

[3] Wired network can have multiple network interfaces too but they are less problematic because connections usually change less frequently.

it is in sleep mode with its network interface turned off. The initial assumption of nodes being always connected is no longer valid in such networks.

In order to overcome this problem for nodes wishing to communicate with each other, it is necessary to determine common periods in which they are turned on, a method referred to as rendezvous [44]. Typically, nodes would follow predetermined on/off-periods. They would remain active for a certain amount of time and would then go to sleep until the next on-period. Without explicitly exchanging a schedule, asynchronous protocols allow nodes to wake up on their leisure but can still guarantee rendezvous within a certain amount of time [45].

Later, WSNs have been extended to operate on more than one channel, giving rise to the development of so-called multi-channel MAC protocols. As sending and receiving node could potentially be tuned to different channels, a multi-channel rendezvous protocol was needed to determine common channels which both sender and receiver are tuned to when communication is taking place [46, 47].

But wouldn't that mean that such a mechanism, regardless of whether using a single channel or multiple channels, would be a new link layer function that needs to be provided next to all other functions? Even though it may be argued that energy efficiency, as such, is a non-functional requirement, it has an impact on various link layer aspects, for example on the wakeup and scheduling policy of a MAC. In operating system research, for example, there are approaches that treat energy as just another system resource [48]. Similar to that, it may be also viewed as an important system aspect or resource that needs to be managed and controlled through the protocols of a communication system. JTP [49, 50], for example, successfully demonstrated its potential for energy saving by carefully adjusting and balancing various protocol mechanisms, such as congestion avoidance and error recovery.

2.3.6 Summary

The main task of the link layer in a communication system is providing a means for exchanging frames over the physical network. We have highlighted the three basic functions of a link layer protocol and discussed MAC as the fourth, which is necessary when broadcast channels are used instead of point-to-point links.

The commercial launch of wireless networks has certainly been a big step towards better quality and service for users of personal communication systems. From an overall link layer point of view, however, the core functionalities and responsibilities largely remained the same.

In contrast, the advance of mobile and multi-homed networks introduced a number of new challenges that also need to be addressed at the link layer. However, how to solve them in an adequate manner is still subject of intensive research.

A further question that also remains unanswered is that of how the link layer in DSA systems differs from traditional communication systems? The next section will investigate this issue by studying the requirements of link layer protocols for DSA.

2.4 Link Layer Functions of Future DSA Systems

The link layer undoubtedly plays an important role in several of the features expected by FWCS. This includes new ways of accessing frequency spectrum as well as new ways of communication

between devices. But how those functions influence the development of link layer protocols or protocol architectures largely remains an open issue. What are the core differences of the link layer in future DSA systems to those in conventional systems? Are there any new requirements? How can they be accommodated in a practical system?

Before those question can be addressed in Chapter 4, we have to discuss the new requirements that the link layer of FWCS has to cater for. From this point on, the discussion focuses on DSA systems as one technology component of FWCS, if not explicitly stated otherwise. We start this discussion by briefly analyzing the changes that arose through a non-static, reconfigurable PHY layer. We then elaborate on the necessary changes to the link layer for enabling DSA, both from a functional and a non-functional perspective.

2.4.1 Managing Physical Layer Resources and Capabilities

Conventional communication systems are designed with specific scenarios in mind. When following the spirit of CR, however, FWCS are viewed as devices that are capable to adapt themselves to the current operating conditions in order to best serve their users in a multitude of application scenarios.

For this purpose, the PHY layer exposes a multitude of meters and knobs that may be used to observe the radio's performance and to adjust its operating parameters, respectively. Literally all parameters of the radio, be it the actual air interface to use, its frequency, bandwidth, antenna, beam shape, modulation type, medium access strategy, frame structure, or queuing policy, are subject to change and have to be viewed in the context of the radio system that tries to provide highly reliable communication while efficiently utilizing the available resources[4].

Therefore, managing the services, resources, and capabilities offered by a flexible PHY layer becomes an important link layer aspect in FWCS. This includes the entire operating environment of a radio and may involve control and coordination of spectrum sensing, enforcement of radio policies and operating restrictions, and the coordination of user and application requirements.

2.4.2 Supporting Link Establishment

Before nodes can exchange information with each other in a communication network, they need to establish links among themselves. In traditional (wired) networks, this link establishment phase is often quite simple, as network nodes are usually expected to be running and connected to the network[5] all the time. Therefore, if a client wants to connect to a server or to another client in a wired network, it would just use its networking interface to reach the intended peer. Because the server is never turned off, the client can be sure that its message can be received without prior coordination. This is certainly a simplified description that doesn't include resolving names or addresses, multi-hop communication or packet loss which would be required in most realistic scenarios, but serves well to show that nodes can be considered to be *always connected*.

With the development of wireless networks, such as IEEE 802.11 [51], the situation changed only little. They are usually operated in infrastructure mode which means that a central coordinator or AP exists in the network that coordinates a set of client nodes. An AP operates on a fixed

[4] A comprehensive table of possible radio parameters can be found in [30, p. 227].

[5] A node may have one or more network interfaces but all of them are connected during normal operation

channel available to all users. In order to allow clients to identify this channel, the AP periodically broadcasts beacon signals. Once all nodes have found the channel and are associated to the AP, communication is similar to wired networks.

With the advance of large-scale mobile networks such as WSNs, this situation changed drastically, as has been discussed above. But what exactly differentiates them from DSA networks?

In DSA networks, the ever-changing environment, PU activity, and the geographical separation of nodes may lead to varying channels conditions. Without even knowing the existence of each other, link establishment in the context of DSA also has the important responsibility to create self awareness among network nodes. Naturally, being aware of the presence of a peer is essential in order to set up a link, to exchange information, to manage available resources, and to communicate data. Therefore, supporting nodes in initially setting up links among each other presents a crucial link layer aspect in FWCS. This especially holds true in unknown environments.

Throughout the thesis, we will use the term *link establishment* to refer to the general mechanism for setting up links among nodes in a network. However, most algorithms in the context of CR have been introduced as *rendezvous* algorithms. As this term seems to be well-established in the field, we will also use it for referring to a specific algorithm. It should also be noted that in the CR field, they are often used synonymously [52, 53], sometimes causing misunderstanding between other variants of rendezvous.

It is also noteworthy to say that some kind of link establishment is also required in future LTE generations, like *LTE Direct*, which is a device-to-device communication technology. In this context, link establishment is also known as "device or proximity discovery" [54].

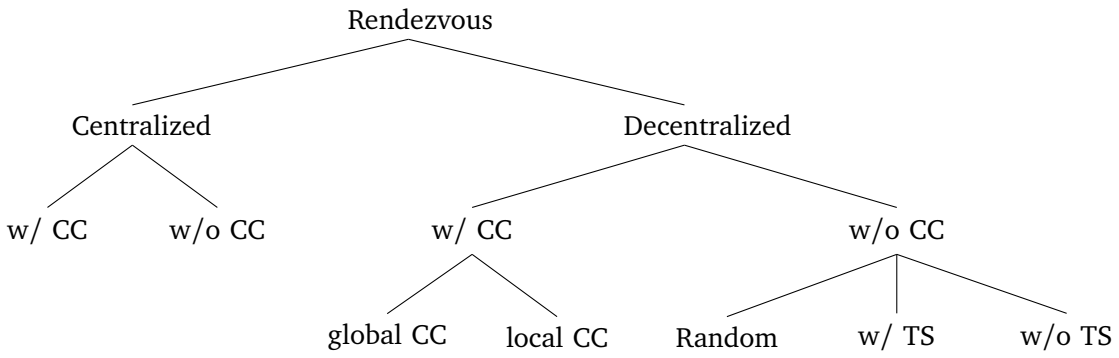
2.4.2.1 Algorithm Classification

The actual implementation of a link establishment mechanism greatly depends on the application and network scenario under consideration. Networks may be heterogeneous like cellular networks or homogeneous, comprising only one type of node. The applicability of a given algorithm also depends on the channels that are available to the users.

In the literature, different classification schemes for rendezvous algorithms have been proposed [52, 55]. Following Liu *et al.* [55], they may be categorized into *centralized* and *decentralized* algorithms. Centralized networks typically consist of heterogeneous nodes. Therefore, a dedicated controller (i.e., the leader) coordinates and guides ordinary nodes (i.e., the followers) through the process of link establishment. Contrary to that, nodes in a decentralized network are typically of the same kind. Figure 2.5 graphically illustrates this classification scheme. In the remaining part of this section, we will further analyze both centralized and decentralized algorithms.

Centralized Algorithms

Centralized algorithms often employ a so-called control channel (CC) that aims at simplifying the establishment procedure. Generally speaking, a CC is a channel that is assumed to be available to either all users in the network (global CC) or to a set of nodes (local CC). It can be either static or dynamic. The term CC has been introduced to differentiate between channels being used to transfer either control information or data. This differentiation is extremely helpful if a single protocol fulfills two tasks in parallel and uses different channels for this purpose. From



Definitions:
CC Control channel
TS Time synchronization

Figure 2.5: *Rendezvous algorithm classification (reproduced from [55]).*

the viewpoint of a link establishment protocol, however, the separation between control and data channel is usually not required. We therefore only differentiate between static or dynamic channels.

Although the use of static control channels can simplify link establishment significantly, they are often impractical and undesirable due to various reasons:

- **Availability:** Due to a dynamically changing environment, the availability of channels varies with time and space and can therefore not be guaranteed. A pre-selected channel may therefore not be available in a certain area at all, or, may be occupied by another user at certain times.

- **Congestion:** Even if a static CC would exist, it is likely to suffer from severe congestion and high collision rates, especially if the number of nodes increases and the entire control information is transfered over a single channel.

- **Robustness:** A single channel is vulnerable to jamming attacks. Being a single point of failure, it would interrupt the entire network operation if it fails.

The choice of a suitable link establishment algorithm should therefore always consider whether it operates on static or dynamic channels. In general, both centralized and decentralized algorithms could be realized with both types. However, it should be noted that even if a centralized algorithm operates on dynamic channels, the central controller itself remains as a single point of failure, thus affecting the overall robustness of the system. Representatives of the centralized protocol family operating on dynamic channels are *Exhaustive Search* (EX) [56] and *Deterministic Approach to Rendezvous Channel* (DARC) [57]. The cognitive pilot channel (CPC) approach [58] also assumes a centralized architecture, but relies on a CC. The underlying idea and details of EX will be explained in Section 3.2.

Decentralized Algorithms

The second branch of Figure 2.5 contains algorithms for decentralized scenarios. Here again, two subcategories are possible. The first category contains algorithms that require a CC and the second category those that don't require a CC. Needless to say that the same restrictions in terms of availability, congestion, and robustness are also valid in the decentralized case.

Decentralized algorithms that use dynamic channels are also called *blind-rendezvous* algorithms [52]. They are called blind because they can neither assume the availability of certain channels, nor the existence of a centralized controller to assist the rendezvous procedure. From a researcher's point of view, this is the most challenging category and, thus, has attracted a lot of scientists over the past years.

The majority of available algorithms are based on channel hopping (CH). In CH-based solutions, each node, equipped with a single radio transceiver, hops over a pre-defined sequence of channels until a link is established. This thesis only focuses on CH algorithms because they are closely related to the link layer and can be realized independently from the underlying PHY. However, it is noteworthy to say that the research community has also brought up PHY layer approaches to this problem [59, 60]. In their current form, however, they also assume a centralized environment with distinct leader/follower roles.

Among the most prominent CH algorithms in the decentralized category are *Modified Modular Clock* (MMC) [52], *Channel Rendezvous Sequence* (CRSEQ) [61], and *Jump and Stay* (JS) [53, 62].

Properties that could further help categorizing algorithms include whether they require time synchronization (TS), the channel model that they are applicable in (i.e., symmetric or asymmetric), the amount of nodes they support (i.e., only two or more than two nodes), the amount of radio transceivers available to each user, and whether they can provide an upper bound for rendezvous. In our analysis, we narrow the scope and solely focus on algorithms that require a single radio transceiver only.

As has been stated above, the algorithmic details will be explained in Section 3.2. Selected algorithms are then further studied and analyzed through simulations in Chapter 5.

2.4.2.2 Performance Metrics

Rendezvous algorithms are typically evaluated in terms of the time needed to establish a link. The fundamental metric is called Time To Rendezvous (TTR). TTR is defined as the number of slots "that it takes for two or more radios to rendezvous, once all radios have begun the rendezvous process" [52]. The Maximum Time To Rendezvous (MTTR) is the worst case time of an algorithm. If the MTTR of a given algorithm has an upper bound, it is said to achieve *guaranteed rendezvous*. We will use both metrics later on in this thesis to evaluate selected algorithms in both centralized and decentralized scenarios.

2.4.2.3 Summary

This section argued that supporting link establishment is an additional function that needs to be addressed in FWCS. We have highlighted the main challenges and have presented a classification of possible approaches to the problem. At this point, even without analyzing the above mentioned algorithms in detail, it can already be seen by the number of classification criteria that a single

strategy implemented in a single protocol may not be enough to serve the multitude of possible scenarios and network conditions. For supporting a wide range of possible use cases, it is therefore desirable to have a flexible protocol architecture that can be configured according to the demands of a specific application.

As has already been stated above, we will revisit and explain some of the algorithms mentioned in this section in the next chapter. Furthermore, we will implement a selected set of algorithms and compare their performance quantitatively through simulations as well as through practical experiments in Chapter 5. Before that, we need to discuss an additional link layer function that also needs to be addressed in FWCS.

2.4.3 Supporting Spectrum Mobility

As has been discussed in the previous section, mobility in traditional mobile networks can be viewed as dynamic multi-homing. In essence, what this means is that if somebody walks around, let's say through an urban area of a larger city, his or her phone constantly connects and disconnects to different base stations while moving into the coverage area or out of coverage, respectively. A phone that is connected to more than one base station at a time can formally be described as a multi-homed mobile user. This functionality is controlled by the cellular network that takes the required measures in order to guarantee seamless operation. But what happens in DSA networks? Can we draw a similar analogy? Can we compare spectrum mobility caused by PU networks or other SUs to mobility in traditional networks?

In order to be able to provide scientifically grounded answers in Chapter 5, the remaining part of this section introduces the required basic knowledge. In the following, we discuss spectrum mobility in DSA systems and mention possible handover strategies and performance metrics. This discussion is based on Christian *et al.* [63].

2.4.3.1 Spectrum Handover Cycle

The process of suspending an ongoing transmission, vacating the channel and resuming the communication on another channel is called spectrum handover. A spectrum handover may take place due to various reasons. In DSA networks, a spectrum handover may be required due to the sudden appearance of the PU when operating on a licensed channel or when entering the service area of a PU network. Independently from the channel access policy, a spectrum handover may also be required due to environmental changes causing degraded communication performance. In such a case, moving an ongoing communication onto another channel, e.g., to another frequency band, may be sufficient in order to provide the required services. As the communication environment is constantly subject to changes, especially in mobile scenarios, FWCS are required to develop some kind of awareness that allows them to react to those changes. The entire spectrum handover can be understood as a two-stage process that consists of an evaluation phase and a link maintenance phase. During the evaluation phase, a radio node observes and analyzes its environment. Once a handover decision has been made, it enters the maintenance phase and suspends any ongoing communication. It then performs the handover, i.e., tunes to the target channel, after which it resumes the communication again. The node then reenters the evaluation phase and the cycle begins again.

If an error occurs during the handover, for example because the selected target channel is occupied by another user, the radio can either select a new target channel based on the available information or it can transit into the evaluation phase and restart sensing. Figure 2.6 depicts the so-called spectrum handover cycle.

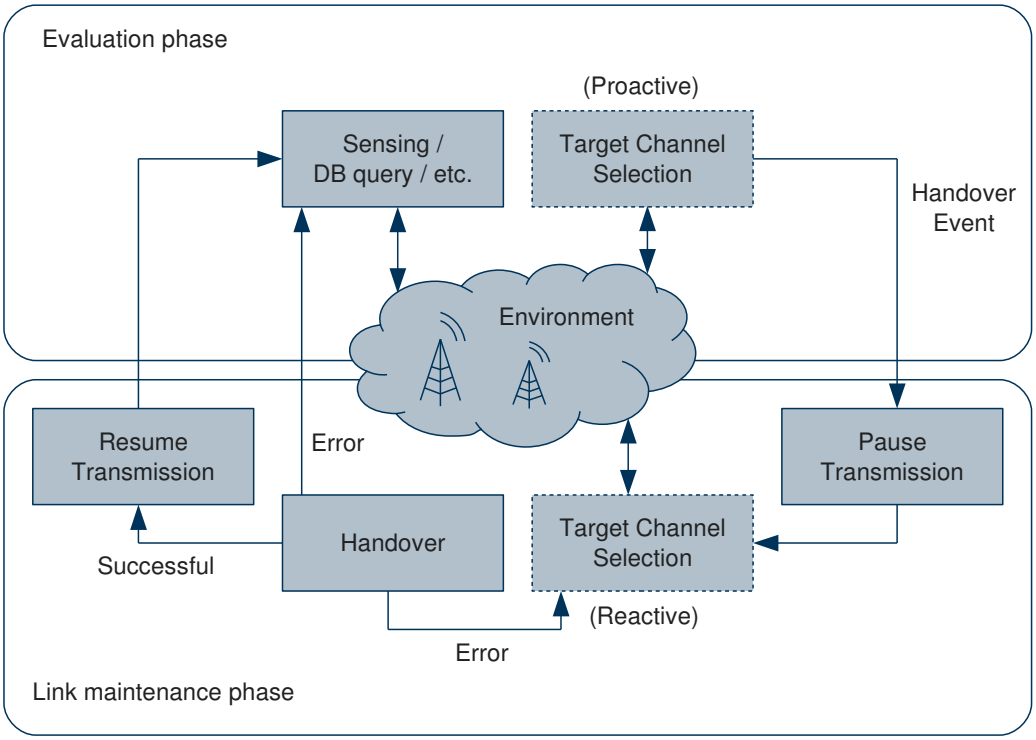


Figure 2.6: *Spectrum handover cycle (reproduced from [63]).*

2.4.3.2 Target Channel Selection

The process of selecting a suitable handover target channel is a non-trivial problem addressed by many researchers and engineers. For answering this question, one typically has to study numerous factors that are influenced by the operating environment. This includes the type of network nodes, their activity, general channel availability, and local performance requirements, such as the minimum tolerable interference time.

Information about potential target channels can be either obtained through spectrum sensing [64], statistical models [63], or beacons and databases [65].

2.4.3.3 Handover Strategies

Spectrum mobility may be categorized into four strategies, according to the time when target channel search and spectrum handover take place. Both can happen either before (proactive) or after (reactive) the event that has caused the spectrum handover to occur.

- **No handover:** Using this strategy, the SU simply suspends communication and stays in the channel as long as the PU is active. The SU then resumes after the PU has disappeared again. Understandably, this strategy may cause high communication delays if the PU is active for a long time. On the other hand, the costs for target channel selection, negotiation, and handover can be avoided. In [66], it has been shown that in some scenarios, connection buffering, as the paper calls this strategy, performs better than other strategies due to the channel switching overhead.

- **Pure reactive handover:** Using the pure reactive strategy, the SU performs both target channel selection and handover after the trigger event has occurred. Although this strategy may get more accurate results, this may come at the cost of an increased handover delay due to on-demand spectrum sensing or database queries.

- **Pure proactive handover:** Using the pure proactive strategy, the SU selects a target channel and vacates its current operating channel before the PU occupies it. This can be achieved through PU activity prediction or PU schedules that may be available in a database. The advantage of this approach is that the handover can be planned in advance, which may reduce the length of the interruption. On the other hand, if the PU activity model is inaccurate, spectrum handovers may disturb communication unnecessarily.

- **Hybrid handover:** Hybrid handover is a combination of pure reactive and pure proactive handover. Usually, the selection of the target channel is done before the handover event occurs. The actual handover, however, is done after the event.

The question which particular handover strategy is more efficient depends on the characteristics of a given network environment. This include the amount of a priori knowledge available about the PU as well as the costs for sensing, target channel selection, and handover.

In highly loaded networks with only short PU activity, the no handover strategy may be the best, as handover costs can be completely avoided. In contrast, the pure proactive strategy may be preferred if detailed knowledge about the PU activity is available. Pure reactive and hybrid strategies may be considered the general-purpose concepts, as they can be applied in most scenarios with acceptable performance.

2.4.3.4 Performance Metrics

Possible metrics for assessing the performance of spectrum mobility solutions include:

- the number of handovers during a specific amount of time,

- link maintenance probability, and

- handover latency.

Link maintenance probability refers to the likelihood that a link is seamlessly maintained over the entire handover procedure. In other words, if the probability of link maintenance is 100%, each handover is successful and not a single connection gets lost or interrupted during handover.

Clearly, the more handovers are needed, the more time is spent that can't be used for actual communication. On the other hand, more handovers may decrease the link maintenance probability [67].

From the PU's point of view, handover latency may be the most relevant performance metric, as it has an impact on the amount of interference caused to the PU network.

2.4.3.5 Summary

This section has studied the problem of spectrum mobility in FWCS. Our analysis has again shown that it is difficult to determine a single strategy that leads to an optimal solution in any given communication network. Even if this optimal solution can be found for a particular scenario, it may only be valid for a bounded amount of time due to rapid changes of the environment.

We therefore argue that a single handover strategy implemented in a single link layer protocol generally can't lead to satisfactory performance. Instead, a flexible protocol architecture is required that can combine multiple protocols with each other, according to the demands of the communication system. Ideally, adaptive solutions that are able to detect and select the most suitable handover strategy for a particular scenario are required for the best possible performance.

2.4.4 Architectural Requirements

This section describes the generic, non-functional requirements that serve as the basis for both evaluating related work and building the FLL architecture proposed in this thesis.

Motivated by the desire to support a multitude of applications, the core non-functional requirements of the link layer in FWCS can be summarized as follows:

- **Flexibility:** Flexibility is the ability to "be changed (and) to respond to different circumstances" [68]. This implies that the architecture should not be limited to a single application, but should be capable to work under varying requirements and conditions. In the context of this thesis, we further differentiate between two forms of flexibility that are closely related to each other: **adaptability** and **extensibility**. Adaptability describes the ability to "become adjusted to new conditions" [68], i.e., to adapt to changing characteristics of a communication link during *runtime*. In contrast, extensibility describes the capability to extend or enrich an existing system by a new, possibly unforeseen function, for example by adding a new protocol. This extension does not need to be carried out during runtime necessarily.

- **Reusability:** In engineering, reusability means the ability that existing protocols or parts of them can be used again in new protocols with minor modification or no modification at all [69, p. 193f]. The main intention behind reusing existing components, rather than handcrafting new ones, is to achieve better quality through well-tested implementations as well as to decrease development time and costs.

- **Complexity reduction:** Given the multitude of functions that need to be carried out at the link layer, it is desirable to reduce the complexity of protocols in order to help understanding and maintaining them.

2.4.5 Summary

This section has studied the extended set of functions that the link layer of FWCS is expected to handle. We have found that a dynamic and reconfigurable PHY layer poses a significant change in the link layer compared to traditional networks. This is because the characteristics of the entire PHY layer may be subject to change and, thus, must be managed and controlled. We have also discussed the need to support link establishment and spectrum mobility as a link layer function in order to realize DSA networks.

What now comes to mind is the question how these new functions can be accommodated in an efficient manner? They shouldn't increase the complexity of the architecture unnecessarily and basic functions shouldn't be sacrificed either. An intuitive solution to this problem is the integration of, for instance, a link establishment or spectrum mobility mechanism into existing protocols. While at the first glance it is tempting to do so, there are good reasons for keeping them separated. First and foremost because the protocols are doing very different things. While the core link layer is responsible for providing a communication channel between nodes (i.e., providing data transfer services), a mobility protocol is rather a management protocol.

Furthermore, heterogeneous system requirements may demand specialized protocols for specific functions. Such protocols could clearly benefit from reusable core blocks that implement common functionality shared among different protocols. Therefore, the link layer architecture of a FWCS should be flexible in the sense that it supports all those requirements and, thus, facilitates the development of novel protocols.

2.5 Practical Link Layer Implementation

This section discusses different possibilities for practical link layer research and implementation using both conventional wireless NICs as well as SDRs. We will first highlight and briefly discuss the main implementation challenges when using SDRs. After that, we will introduce *Iris*, the SDR framework that is employed in this dissertation for practical experimentation.

2.5.1 The Latency Challenge

The vast majority of experimental work on SDR platforms has dealt with the PHY layer. Implementation-specific issues, in general, are well understood, even though ever-increasing sampling rates will always remain demanding. Work on higher layers of the protocol stack, however, imposes significant research and engineering challenges for another reason. Processing delays as well as latencies caused by bus and queuing delays are particularly challenging, especially when it comes to the implementation of MAC protocols [35]. This is primarily because many protocols have strict timing requirements that need to be met in order to guarantee flawless operation.

Let's consider IEEE 802.11 as an example. The distributed coordination function (DCF) [51] defines a number of time constants for the PHY and MAC layer which need to be met by all participants of a network. These constants, including slot time, short interframe space (SIFS), and DCF interframe space (DIFS), are tightly connected with each other. They depend on air propagation time as well as on PHY and MAC processing delays.

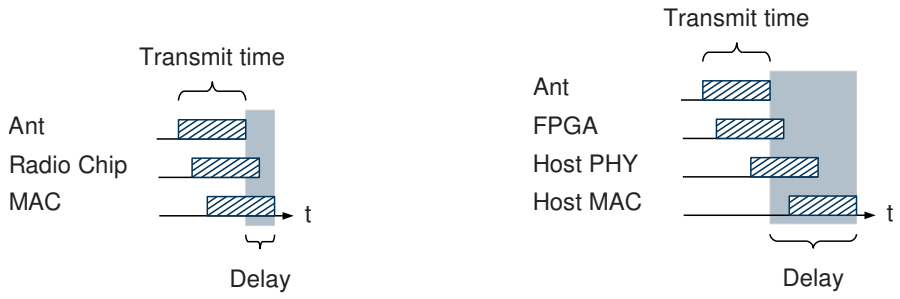
To reduce the probability of frame collisions during communication, nodes that have data to send first have to contend for the channel. This is usually achieved through a mechanism called clear channel assessment (CCA), which needs to be executed prior to sending a frame. Only if the medium is found to be idle for a certain amount of time, the DIFS period, the node starts transmitting the frame. If the medium is found to be busy, the node backs off and doesn't start transmitting immediately, in order to avoid a collision on the shared medium.

If the communication delay between PHY layer, which performs the CCA, and MAC layer, which implements the transmission mechanism, is too high, the channel state could have changed already by the time the CCA result is evaluated.

Figure 2.7 compares the packet processing delay (without correct scaling however) in conventional wireless communication systems and host-based SDRs. Host-based SDRs have already been introduced in Section 2.2.3. In a nutshell, they are characterized by the fact that they implement the majority of their DSP tasks on a commodity host computer in software. As a consequence, bus latencies between digital radio backend and host-based DSP blocks, and the low computational power of general purpose processors may introduce significant delays in host-based SDR architectures.

In contrast to that, the delay between receiving a signal at the air interface and processing the corresponding packet at the MAC layer in conventional IEEE 802.11 NICs is typically in the order of microseconds. This is due to a highly optimized application-specific and integrated hardware/software design. Figure 2.7a illustrates the "antenna to MAC delay" in such a conventional radio.

Figure 2.7b illustrates the impact of a host-based SDR on the CCA mechanism employed in CSMA protocols. In this example, the sum of all individual delays exceeds the transmit time of the packet. Hence, the channel state may be outdated by the time MAC receives the result of the CCA cycle. Therefore, the CCA mechanism can only work if the actual transmit time is significantly larger than the processing delay. This example demonstrates how tightly PHY and MAC layer are interconnected with each other. Even though host-based SDR architectures exhibit significantly worse delay characteristics, compared to conventional IEEE 802.11 NICs for example, it is still feasible to realize a CSMA-MAC, as will be shown later.



(a) *Conventional radio.* **(b)** *Host-based SDR.*

Figure 2.7: *Antenna to MAC delay in conventional radios and SDRs (derived from [34]).*

2.5.2 Design Space

In the previous section, we have discussed the latency issue of host-based SDRs and compared them to conventional off-the-shelf wireless NICs. Both concepts are near the extreme edge of the possible design space. We will now explore general options for designing and implementing link layer protocols and discuss their advantages and disadvantages. The possibilities for implementing link layer protocols can be broadly categorized into three subgroups:

- hardware-based,
- software-based, and
- hybrid.

We will now discuss each subgroup separately. Afterwards, we will compare them qualitatively.

2.5.2.1 Hardware-based

Hardware-based solutions can be further categorized into static or reconfigurable. Static hardware consists of highly optimized circuits, such as an application-specific integrated circuit (ASIC), that cannot be modified after fabrication. In contrast to that, reconfigurable hardware, such as an FPGA or a DSP, can be reprogrammed after manufacturing. The main advantage of hardware-based solutions is their high computational power and deterministic execution time.

Off-the-shelf IEEE 802.11 NICs are typical examples for cheap, mass-produced devices. Although the research community has widely used them to implement various protocols that go far beyond the intended usage (see [70, 71, 72]), the degree of flexibility they offer is limited. Ultimately, the manufacturer of the device defines the boundaries.

FPGA-based solutions do not have such a limitation because their hardware design is modifiable. From the performance point of view, they offer almost the same advantages as static hardware, but are more expensive in terms of cost and power consumption. The *Wireless Open-Access Research Platform* (WARP) [73, 74], developed at Rice University originally, is widely used in the research community for PHY and MAC layer development. Note that we categorize the WARP platform as a hardware-based approach because the entire PHY layer and the time-critical part of the MAC layer are implemented in hardware [75]. This is in contrast to other research platforms that leave the majority of the DSP tasks on the host [35, 76, 11].

A disadvantage of FPGA-based solutions is the limited amount of physical resources on a given type of FPGA. This includes lookup tables, signal processing, and memory blocks. Especially memory can be a limiting factor when implementing protocols that require frame buffering. A possible example are QoS-aware protocols with transmission queues for multiple traffic classes, which reduces the degree of flexibility. Moreover, and probably the most important factor for researchers, the development of new designs and modifications to existing ones is generally considered to be more expensive in terms of cost and time compared to software-based DSP [30].

2.5.2.2 Software-based

In contrast to hardware-based solutions, software offers the highest degree of flexibility. Modifications can be done in a short amount of time and memory is usually not constrained. However,

the limiting factor of software in combination with general-purpose hardware when it comes to radio applications is their DSP performance itself and unpredictable processing time [77]. This is primarily due to their general-purpose character, which includes the hardware, the operating system, and its surrounding ecosystem that are not optimized for a single use case, but targeted for a wide application range.

Nonetheless, it has been shown by Tan *et al.* [78, 79] that by carefully designing and optimizing the entire system, even a software implementation of a standard-compliant IEEE 802.11 transceiver is technically feasible. However, it is still fair to say that this wouldn't have been possible without highly optimizing the entire signal processing chain. This includes designing a custom, low latency PCI Express RF front end as well as intensively using hand-optimized assembly code and lookup tables.

2.5.2.3 Hybrid

The third possibility for implementing link layer protocols is a combination of the hardware- and software-based approach, with the main goal to combine the advantages of both strategies. Time-critical aspects are implemented in hardware in order to achieve deterministic behavior whereas most of the non-time-critical system parts remain in software for higher flexibility. This concept is also called hardware-software co-design.

The main disadvantage of the hybrid approach is its complexity due to the split of responsibilities between hardware and software components that require a careful design process. Particular care must be taken when designing possible interactions between components such as feedback mechanisms.

The "split-functionality architecture", developed by Nychis *et al.* [35], was among the first functional implementations of the hybrid concept. The prototype featured a *GNU Radio* (GR) implementation of a CSMA-MAC that realizes CCA, backoff, and acknowledgement processing inside the FPGA of a first generation USRP [35].

Later, Di Francesco *et al.* [80, 76] implemented a carrier sensing block inside the FPGA of the embedded version of the USRP (USRP-E100). Even though implemented inside an FPGA, the USRP-E100 solution was still not able to meet the IEEE 802.11 timing requirements.

Very recently, Bloessl *et al.* [81, 11] have developed a standard-compliant transceiver for IEEE 802.11 broadcast transmissions based on GR.

2.5.3 The Iris Software Radio Architecture

Iris is a software framework for designing reconfigurable, component-based SDRs [82, 83]. Essentially, it is a framework for creating a graph of components connected via links. Those links connect the input and output ports of their respective components. Components can communicate with each other through messages that are passed along the links. Within *Iris*, XML files are used to define the components and their links in a formal way. Upon execution of *Iris*, the core software reads the XML, constructs the flow graph, loads the corresponding components, and executes the radio. *Iris* defines three types of building blocks that may be used to describe a radio. They can be summarized as follows:

- **Components:** Components are self-contained entities for implementing a discrete radio

function, such as a filter, modulator, or even an entire link layer protocol. They are characterized by a set of input and/or output ports which are used to connect them with one another.

- **Engines:** Engines are an abstract concept to encapsulate one or more components within a radio. In fact, an engine defines how the specific part of the flow graph under its regime is executed, how data is passed between components, and how the reconfiguration is organized. A radio may consist of one or more engines. Two types of engines are defined: *PHY engines* and *Stack engines*. As the name suggests, a *PHY engine* is best suited to implement the PHY layer of a radio. A single thread of execution sequentially calls all components within the engine according to the data flow defined in the radio configuration. The execution of a *PHY engine* may be compared to the stream-oriented processing of GR.[6] In contrast to that, the *Stack engine* allows developing higher layer protocols, such as a link or network layer protocol. They are characterized by the fact that they operate on packet granularity rather than on a stream of bits. Moreover, they are likely to consume and produce packets at the same time and have a bi-directional data flow. In contrast to a *PHY engine*, each component of a *Stack engine* runs in its own thread of execution.

- **Controllers:** In *Iris'* terminology, a controller is a radio block that is capable to reconfigure all component parameters as well as to change the structure of the entire radio by inserting or deleting components and links. Controller activity is usually triggered by events that may be sent out by other components or controllers.

2.5.4 Summary

In this section, we have highlighted the core challenge of practically implementing link layer protocols in wireless communication systems: latency. We have then discussed possible choices for designing and implementing link layer protocols using SDRs. All of them have their advantages and disadvantages. Table 2.1 qualitatively summarizes the different concepts.

The flexibility offered by software solutions in their purest form usually comes at the cost of increased processing and computation time. Static hardware, on the contrary, should only be used if the degree of flexibility offered by the hardware is guaranteed to be sufficient for the intended purpose, as any modification (of the hardware) is impossible. The dynamic hardware approach, e.g., a pure FPGA implementation, should be taken if performance is of primary importance and the intended design is relatively mature, as the entire development process as well as major modifications may be time and cost intensive.

Although a pure software implementation offers the highest degree of flexibility and, thus, can be extremely useful for protocol prototyping and experimentation, it fails in meeting strict temporal requirements. It is therefore suitable for data transfer protocols but disadvantageous for many MAC designs.

The hybrid approach combines high flexibility and deterministic behavior at the cost of a higher implementation complexity. This approach provides a good trade-off when reconfigurability and performance are equally important. Thus, it can be used for both data transfer protocol and MAC implementation. As a consequence, this dissertation employs both software-based and hybrid

[6] Even though scheduling in GR works slightly different because each GR block runs in its own thread of execution.

Table 2.1: *Summary of possible choices for experimental link layer implementation.*

	Static hardware	Dynamic hardware	Software	Hybrid
Costs	low	high	low to medium	medium to high
Flexibility	low	medium	high	high
Modification costs	-	high	low	medium to high
Complexity	low	medium	low	medium to high
Deterministic	yes	yes	no	yes
Example system	Atheros chipsets (e.g., AR2425)	WARP	GR/*Iris* for PHY and MAC	GR/*Iris* for PHY, USRP for MAC

solutions for practically implementing its envisioned concepts. In all practical experiments, *Iris* is employed as the underlying SDR framework.

2.6 Software and Protocol Engineering Principles

In this thesis, we explore the realization of a software architecture for implementing flexible link layer protocols. This section briefly discusses three fundamental engineering principles that are key for achieving the above stated goals. The first two principles are related to software engineering in general. The third principle is rather specific for protocol engineering.

2.6.1 Componentization

Componentization, or modularization, is a design concept that aims at dividing a large system design into smaller, manageable components; a typical "divide and conquer" approach [84, 69]. Two primary objectives are followed: *weak coupling* and *strong cohesion*. Coupling refers to the degree to which components depend on one another. It allows to distribute labor among different workers or classes with a minimum amount of interaction among them. In contrast, "cohesion refers to the degree to which module components belong together" [85]. Strong cohesion allows a developer to design a specialized solution for a specific problem.

As a consequence, within an architecture with low coupling, it should be easier to include a new component, for example by implementing a new protocol. Within an architecture with high cohesion, it should be easier to implement new protocols without being concerned with aspects that are not directly related to the aspect of interest.

2.6.2 Separation of Concerns

The separation of concerns (SoC) concept postulates the separation of different functional aspects (called concerns) into different components, "such that each piece corresponds to a semantically coherent issue of the problem domain of interest" [86]. The term itself has been coined by Dijkstra

in 1974 [87]: "even if not perfectly possible, is yet the only available technique for effective ordering of one's thoughts". The "goal (behind the SoC concept) is not to reduce a system into its indivisible parts, but to organize the system into elements of non-repeating sets of cohesive responsibilities" [88]. Typically, this can be achieved through decomposition. From an software engineering point of view, there are two possible strategies. How they are applied depends on the abstraction level. On a small scale, one can separate concerns into distinct functions or classes, for example through inheritance. On a large scale, one can decompose a system into parameterizable components. In this thesis, we will explore both approaches.

Unfortunately, however, a clean separation of concerns in a non-trivial system is not always possible. In practice, secondary concerns such as synchronization or QoS may apply to parts or even the entire system, in addition to the core concerns of that system. Such "cross-cutting concerns" may be problematic in some cases as they "may influence the implementation of all of the other system requirements" [69, p. 569].

2.6.3 Separation of Mechanism and Policy

Separation of mechanism and policy is a principle that allows to expand the operating range of a protocol. A communication protocol typically consists of a number of functions, e.g., error control and scheduling. Thereby, "each function is divided into a mechanism and a policy" [38, p. 39]. While mechanisms are static and do not change once the protocol has been specified, the policies applied to them are likely to change. By separating both, one can guarantee that the same protocol can be used by different applications with different requirements. Therefore, this principle may be viewed as the extension of the SoC principle to protocol engineering.

2.7 Summary

This chapter has provided the reader the necessary background knowledge for this thesis. Before discussing our proposed link layer architecture in Chapter 4, we will first survey related work in the context of reconfigurable protocols and protocol architectures, as well as a selected set of specific link layer protocols.

3 Related Work

In the last thirty years, a significant amount of work in the field of link layer protocols has been published. Generally, it can be said that each technology step has brought up novel protocols and architectures as well as new views and insights into the development and design process thereof. Due to the wide scope of the topic, the discussion of related work needs to be limited in its depth and breadth.

The primary interest of this chapter is to understand the relevance and importance of flexibility for communication systems developers and users over the past years. In addition, we are also interested in specific link layer related protocols and optimization algorithms that are of importance for FWCS. As a result, this literature review is split in two categories. We first survey flexible protocols and/or protocol architectures, i.e., work that deals with the design and development of protocols in general. The second part contains a selection of specific protocols in the field of MAC, link establishment, and spectrum mobility, either proposed as entirely new protocols or extensions to existing protocols for solving a specific problem.

3.1 Flexible Protocols and Protocol Architectures

Communication protocols are typically designed to serve a specific use case and to operate in a particular environment. For example, it is widely known that protocols such as UDP or TCP have limitations when used with applications that not exactly match the scenario they were designed for. Video conferencing, for example, would benefit from a transfer protocol that does flow control, but it doesn't necessarily require 100% reliability. Interestingly, however, this seems to be a combination of protocol characteristics that can't be offered, neither by TCP nor UDP [89]. Similarly, it is known that TCP over wireless links suffers from degraded end-to-end performance due to a TCP design assumption being violated in networks with higher bit error rates [90]. This design assumptions leads to low throughput because TCP stops sending frames because frame collisions are misinterpreted as congestion losses.

The desire to support varying application requirements and network conditions with a single protocol have led to the development of frameworks for designing communication protocols that are capable to adapt their characteristics accordingly. To achieve the required adaptability, monolithic designs are replaced by protocols composed of modular building blocks.

Over the past years, multiple composition frameworks at several layers of the protocol stack and varying module granularity have been proposed. In addition to the module (or component) granularity, they can be further categorized into stack- or heap-based architectures, according to how components are connected to each other [91].

Similar to the well-known OSI stack, stack-based architectures have hierarchically layered protocol components. Signaling between components is usually done through metadata that is attached to the front or the back (i.e., header or footer) of the actual data while traveling up or down through the protocol stack. This approach has the drawback that communication across layers may be more expensive than necessary.

In contrast, heap- or graph-based architectures [91], sometimes also referred to as role-based architectures [92], allow components to be wired together in a (theoretically) arbitrary manner. Signaling between components is achieved through events that may be raised independently from the actual data flow. The drawback of this approach is that events are usually dispatched by a central kernel that may be a bottleneck during event processing.

This section provides an overview about flexible protocol architecture proposed by the research community. We begin the review in chronological order by describing earlier results. Earlier efforts mostly aimed at abstracting from the semantics offered from traditional data transfer protocols. We then continue with contributions that have been proposed in the domain of WLANs and WSNs, mostly looking at abstracting from hard-coded protocols and different development platforms, respectively. After that, we discuss work in the context of cellular networks. Lastly, we survey the most recent work that has been carried in the context of CRs and SDRs. Figure 3.1 shows an overview of the related work discussed in this section.

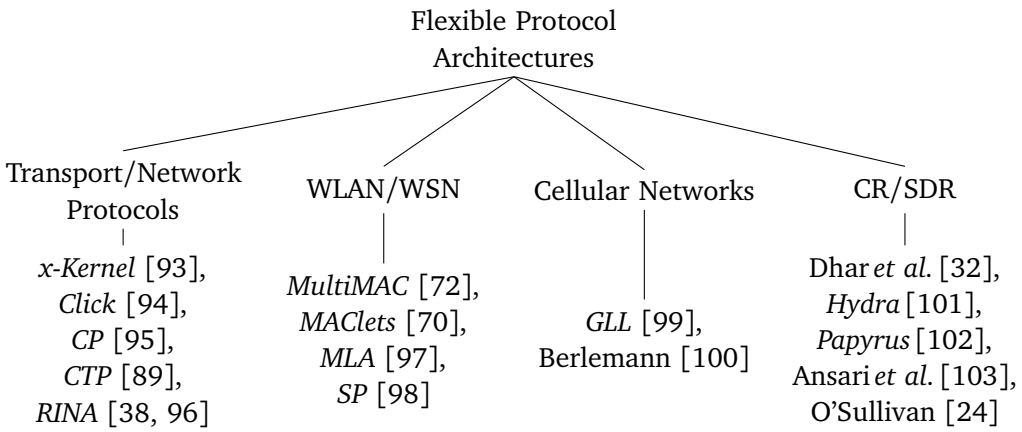


Figure 3.1: *Classification of flexible protocols and protocol architectures according to the field of application.*

3.1.1 In Transport and Network Protocols

The protocols and protocol architectures presented in this section belong to the first attempts in designing flexible communication protocols that mainly dealt with the adaptation of transport protocol semantics to match the specific needs of the application without developing entirely new

protocols. Due to space limitation, we have restricted the amount of proposals discussed in this section. For a more detailed overview, we would like to refer the interested reader to Gazis *et al.* [104].

The x-Kernel: An Architecture for Implementing Network Protocols [93]

One of the first attempts for building hierarchically structured communication protocols was the *x-Kernel*, proposed in 1991 by Hutchinson and Peterson [93]. The *x-Kernel* is described as partly a communication stack and partly an operating system kernel. The architecture differentiates between three communication objects: protocols, sessions, and messages. A protocol is a static object that is defined at compile time of the kernel, it corresponds to a conventional protocol like IP, UDP, or TCP. Protocol objects demultiplex incoming frames from a lower layer and call their corresponding session objects. Session objects are dynamically created as an instance of a protocol. They interpret messages and manage the data structures of the protocol. For instance, a TCP session object would implement the sliding window mechanism including message buffering and retransmission logic. The third type of communication objects are messages. Messages are considered the active objects within the *x-Kernel* architecture. They contain the data and header information of the protocol and travel through the *x-Kernel* up- and downwards.

From a user application point of view, there are several shortcomings of the *x-Kernel*. First, as protocols need to be initialized at boot time, they need to be known in advance and cannot be changed during runtime. Furthermore, the user still has to select a specific protocol, rather than specifying the desired communication characteristics. Nonetheless, there are a number of lessons that can be learned from the work on the *x-Kernel* including the explicit structure of protocols, sessions, and messages, as well as the process/thread per message paradigm that has been followed during the prototype implementation.

The Click Modular Router [94]

Click has been initially proposed by Kohler *et al.* [94] as a modular software architecture for building reconfigurable routers, thus providing evidence that flexibility and reusability also plays an important role for the design of network layer protocols. A *Click* router consists of packet processing modules, called elements, that are bound together as a directed graph. Elements, i.e., the vertices of the graph, implement specific router functions, such as packet classification, queuing, or scheduling. The edges of the graph represent the path on which packets flow through *Click*. From a conceptual point of view, *Click* can be compared to component-based data passing frameworks for SDR, such as *GNU Radio* (GR) or *Iris*. However, one aspect that differentiates it from them is the way how packets are processed and elements are scheduled. In *Click*, packet processing and transfer between elements is based on a push and pull paradigm. On a push connection between two elements, the source initiates the transfer towards the destination element. In contrast, on a pull connection between elements, the destination pulls the frame from the source. Buffering of input and output frames between two elements, if desired, needs to be managed explicitly by a queue element. In GR or *Iris*, buffering between components is managed by the framework itself and is not explicitly visible to the user.

Click has been used successfully in a variety of research projects, including work in the SDR domain, as will be discussed further below.

Design and Implementation of Composite Protocols [95]

Composite Protocols (CPs), proposed by Kannan *et al.* [95], is another attempt for increasing the flexibility and the reusability of traditional communication protocols. CPs consist of single-function components that realize specific protocol features, such as fragmentation or neighbor discovery. They are implemented through finite state machines (FSMs) and arranged linearly in order to provide the required overall functionality of the composite protocol. In order to allow for an easier formal verification of the resulting CP, the sequence of protocol components is decided at configuration time and does not change during operation.

A further building block of CPs is their control interface which provides a mechanism for communication between protocol components. The control interface is modeled through the exchange of messages between a controlled component that offers a service and the controlling component that uses the offered service. One of the main design goals of CPs was to allow formal verification and static code analysis. To evaluate CPs, the authors have implemented a prototype of the *Ensemble* group communication protocol that is written in the functional programming language OCaml.

Overall, CPs represent a very interesting proposal for constructing flexible protocols, in particular in respect of the idea of having FSMs to model protocol components that communicate through messages with one another for realizing some kind of dynamic behavior.

The FLL architecture presented in this thesis leverages some of the CP concepts but applies them at a wider scope in order to address the challenges faced by FWCS, such as link establishment and spectrum mobility.

A Configurable and Extensible Transport Protocol [89]

The *Configurable Transport Protocol* (CTP), proposed by Bridges *et al.* [89], and an earlier version presented in [105], is a modular protocol framework that allows to construct actual protocols out of small components, so called micro protocols (MPs). MPs implement one specific attribute of a protocol. Each application-specific transport protocol is a combination of one or more such MPs. CTP provides a pool of MPs that provide basic functions, such as generating sequenced messages and segments and handshaking for session startup or shutdown. It also provides reliability MPs that provide ARQ and forward error correction (FEC) functionality, as well as transmission control MPs for flow control, ordering, and jitter control.

Interactions among MPs are realized through events. Events can be triggered by other MPs or by external components, for instance to signal the reception of a new frame from a lower layer. The developers have spent an enormous effort to generalize MPs that decouple as much functionality as possible. Naturally, this also led to an increased number of MPs that are required to implement a certain mechanism. This, in turn, led to quite an overhead associated with the processing of events that are needed for handling each and every frame that is received or sent. This issue becomes more severe the more complex the CTP configuration gets, as has been found by the authors. This observation shows that there is a trade-off between flexibility and performance.

Unfortunately, the paper does not mention how CTP interacts with the upper and lower layers, which would be required for interacting with MAC and for managing a flexible PHY layer. Sadly, there is no open-source implementation available that could be used for the purpose of this work.

The Recursive Inter-Network Architecture [38, 96]

Even though proposed in a different field, the research of John Day, and in particular his book "Patterns in Network Architecture" [38], offers valuable insight into the design and implementation of communication protocols. The concepts described in this book have largely influenced the development of the architecture proposed in this thesis.

In his book, Day develops the concept of a *Recursive Inter-Network Architecture* (RINA). RINA is a novel network architecture that is based on the assumption that a network layer is a recurring element with similar functions but different scope. Even though this dissertation doesn't exploit the RINA idea, it leverages some of its useful concepts. For example, the separation of data transfer and management functions is also one of RINA's key ideas [38, p. 202].

In this dissertation, however, the basic SoC concept is further extended to wireless communication systems. Furthermore, we address new issues, such as spectrum mobility and link establishment, which have not been considered by Day.

3.1.2 In Wireless Local Area Networks and Sensor Networks

Commercial WLAN deployments seldom provide optimal performance to their users. This has motivated a great number of researchers to develop experimental platforms for modifying the behavior of the MAC and PHY layer in such networks. Examples of this kind of protocols are *MultiMAC* presented by Doerr *et al.* [72], *FreeMAC* proposed by Sharma [71], or *MAClets*, probably the most advanced approach, presented by Bianchi *et al.* [70] and Gallo *et al.* [106]. Both *FreeMAC* and *MultiMAC* are software solutions that exploit the programmability of certain NIC drivers, e.g., of Atheros NICs. *MAClets*, on the contrary, can be understood as mini firmware updates that are interpreted by a FSM executer inside the hardware device.

Ultimately, however, work on commercial IEEE 802.11 NICs is always limited by the means and degree of reconfigurability exposed by the hardware vendors. It is only offered by a selected set of chipsets and is also limited to the IEEE 802.11 PHY layer standard.

In the domain of WSNs, the diverse range of application scenarios, network deployments, and hardware platforms has led to a large amount of link and network layer protocols that are mostly incompatible with each other [98]. The incompatibility and limited interoperability among different solutions has motivated researchers to propose common abstraction layers that hide details of the underlying hardware.

The *Sensornet Protocol* (SP) for example, proposed by Polastre *et al.* [98], provides a unified neighbor management and message pool and allows to implement different link and networking protocols for sensor networks. Klues *et al.* [97] have further developed this idea and proposed a *MAC Layer Architecture* (MLA) with power management support. The architecture consists of reusable components that implement common MAC features as well as low-level components that abstract details of the underlying sensor node hardware. Although not explicitly mentioned, both architectures also seem to follow a graph-based design with component sizes that lay somewhere between FSM elements and complete protocols. Both are implemented on commercial WSN hardware that is a combination of a relatively static radio part and a programmable microcontroller that contains a software implementation of the protocols and abstraction layers.

From a protocol architecture point of view, the main conclusion that can be drawn from the above two papers is that there exists a clear benefit of abstracting the management related func-

tions of a network, such as neighbor management, from the purely data transfer related functions.

3.1.3 In Cellular Networks

Understandably, generalization and the wish to unify protocol architectures across multiple radio access technologies (RATs) also plays an important role in cellular networks. In this context, Sachs [99] has developed a *Generic Link Layer* (GLL) that "enables a cooperation of different access networks at the link layer". The main motivation behind a GLL in cellular networks is to allow mobile users to seamlessly communicate while moving between multiple networks and RATs. However, this requires a generic layer between the RAT-specific PHY layer and the application layer, hiding the properties of the radio link. The main challenge of such an abstraction layer is the transfer of the transmission context during the handover process between one RAT and the other. Similar to UMTS or LTE, GLL proposes a grouping of link layer function into parameterizable sub layers. Overall, GLL is an interesting approach that aims at harmonizing link layer functions between different RATs in cellular networks. However, although mentioning the need for cooperation and a clear interface between, e.g., mobility management functions and data transfer related protocols, no further details as to how to realize them are given.

A further approach towards modular protocols in cellular networks has been presented by Berlemann [107, 100]. The architecture also addresses multi-mode capable devices, i.e., devices that have to combine multiple RATs. In contrast to the work of Sachs, the flexibility of this architecture is achieved through parameterizable modules that are split into generic and specific parts, i.e., RAT-specific, that are arranged as a graph. Furthermore, the architecture also includes a manager component that administers the layer during runtime as well as a generic interface used by adjacent layers. Although not explicitly mentioned, the architecture doesn't seem to provide a scheduling component to support quality of service for contending applications.

3.1.4 In Cognitive Radio Networks and Software Defined Radios

Work reviewed in this section mostly deals with experimental platforms for developing novel, truly flexible MAC and/or PHY layer approaches in the CR and SDR domain. For this purpose, the majority of approaches employ a SDR framework, such as GR or *Iris*, as their underlaying basis and for implementing the PHY layer. We therefore do not explicitly discuss the frameworks themselves in this section. However, *Iris* is described in detail in Section 2.5.3. For more details about GR, we would like to refer the interested reader to [108].

Supporting Integrated MAC Protocol and PHY Software Development for the USRP SDR [32]

The work carried out by Dhar *et al.* [32] was among the first attempts to exploit the flexibility of SDR for experimenting with flexible MAC protocols. Specifically, the paper proposes a combination of GR for implementing the PHY layer and *Click* for realizing the upper layer part of the radio. The paper also discusses some of the limitations (of the early versions) of GR when it comes to implementing interactive protocols, mainly caused by the sequential processing of the flow graph within GR. For interfacing *Click* and GR, the paper discusses three design alternatives

from which one is used to implement a prototype of the architecture employing a simple TDMA-MAC. Although the achieved performance is quite low due to the large slot lengths, the paper still shows that interfacing different frameworks for SDR-based MAC and PHY layer development is feasible.

Early Results on Hydra: A Flexible MAC/PHY Multihop Testbed [101]

In parallel to the previous work, Mandke *et al.* [101] have also explored the combination of GR and *Click* for implementing flexible protocols in a platform called *Hydra*. In addition to the integration of GR and *Click*, the authors also implement and evaluate an adaptive protocol. This protocol, called *Receiver-based Auto-Rate* (RBAR), is able to adjust the modulation and coding rate at the transmitter based on the signal-to-noise ratio (SNR) measured at the receiver. Though not explicitly targeted for FWCS, we believe the architecture could as well be used to implement spectrum mobility or link establishment in a flexible manner.

Papyrus: A Software Platform for Distributed Dynamic Spectrum Sharing Using SDRs [102]

Another software platform for experimental research on DSA systems using SDR, called *Papyrus*, has been proposed by Yang *et al.* [102]. Essentially, *Papyrus* is a PHY and MAC abstraction layer that consists of a set of API calls for spectrum sensing, i.e., identifying usable frequency bands, and spectrum access, i.e., sending and receiving packets. The prototype of *Papyrus* also uses GR as its underlying PHY layer, while the upper layers are implemented as user-level applications in Python or C++. The paper discusses two examples, a decentralized MAC overlay, called *Jello*, and a PHY layer configuration tool, called *Ganache*. *Ganache* allows to configure the guard bands of the orthogonal frequency-division multiplexing (OFDM) waveform. However, only preliminary results are presented. Therefore, it only serves as a "proof-of-concept verification of the design and implementation" [102].

Conceptually, *Papyrus* is a very interesting initial step towards more experimental research on DSA systems. However, the relatively simple API fails in supporting timing-sensitive MAC protocols. Furthermore, the design misses multiplexing capabilities that would be required to support multiple real applications on top of the MAC layer.

A Flexible MAC Development Framework for Cognitive Radio Systems [103]

Another approach for implementing flexible MAC protocols has been proposed by Ansari *et al.* [103] and related publications, such as by Zhang *et al.* [109]. The papers propose a hardware-software co-design approach which is motivated by the fact that classical MAC protocols implemented in hardware, i.e., conventional IEEE 802.11 cards, only offer limited possibilities for adaptations or reconfigurations.

In order to overcome the limitations of current MACs, the authors have analyzed different protocols, including carrier sense multiple access (CSMA) and TDMA flavors. Based on this analysis, they have decomposed them into functional elements, such as *start timer*, *carrier sense*, *random backoff*, and *send packet*. Based on those elements, they have created a library of FPGA modules.

A tool called the *Wiring Engine* (WE) can then be used to recompose the created blocks into actual protocols.

To evaluate the work, the paper presents a prototype implementation of the framework based on WARP boards. The WE thereby runs on the PowerPC core inside the FPGA of the board, whereas the functional blocks are implemented directly on the FPGA. Although any modification to the FPGA logic requires detailed knowledge in hardware description languages and increases the prototyping time, the co-design approach of the architecture still offers a degree of flexibility that can be compared to software MAC implementations. However, at the current stage of development, the authors have mainly focused on solving purely MAC related issues and have not considered other link layer related topics, such as scheduling or more complex error and flow control protocols.

Ultimately, the idea of implementing time-critical functions inside the FPGA while keeping non-time-critical functions outside the FPGA fabric provides a good trade-off between performance and flexibility. It would therefore be a promising candidate to realize the MAC protocol component of the FLL architecture proposed in this dissertation.

Deconstructing and Reconstructing Medium Access Control: The Anatomy of a Cognitive Radio MAC [24]

Probably the most comprehensive study on MAC protocols in the CR domain, and therefore of high interest to this dissertation, is the Ph.D. thesis of O'Sullivan [24]. Driven by the observation of a missing common understanding with regard to the structure and functionality, O'Sullivan systematically developed a "consistent and structured means of identifying the elements of a CR MAC", called the *Anatomy*.

Essentially, the *Anatomy* is a method that can be used to analyze, classify, and compare existing protocols, as well as to construct new protocols in a systematic and platform-independent manner. Thereby, the *Anatomy* explicitly differentiates between the exploitation and allocation role of a protocol. Exploitation refers to what is traditionally considered the task of a MAC protocol, i.e., allowing devices to communicate with each other over a shared resource. The allocation role refers to a new responsibility that needs to be considered in the CR/DSA domain in order to "avoid excessively interfering with one another", i.e., it deals with allocating and managing communication resources between groups of devices, e.g., a set of APs. In traditional communication systems, this is usually done manually and in a static manner.

This dissertation also builds upon the SoC principle pursued by O'Sullivan, but further extends and advances this concept. Specifically, we assign the exploitation role to two separate components: a generic data transfer and a media-dependent access control protocol. On the contrary, the allocation role, i.e., the management aspect of a protocol, is transfered to one or more isolated components that, similar to ordinary user application, use the data transfer protocol for communicating with one another.

O'Sullivan also highlights the overlap and dependency of MAC and PHY layers that must be considered during the design and development phase. This includes aspects that are specific to the implementation platform, e.g., deterministic delays, as well as general responsibilities that must be satisfied, such as the access to and control of PHY features like bandwidth, frequency, modulation, and power. Within the *Anatomy*, they are explicitly modeled through PHY dependencies and contracts. O'Sullivan also discusses a prototype CR MAC, called *CycloMAC*, that is

designed using the *Anatomy* scheme and implemented on the basis of *Iris*.

The protocol uses a TDMA-MAC with a decentralized synchronization mechanism. Furthermore, it employs cyclostationary signatures [59] for link establishment and spectrum mobility. Strictly speaking, however, this is a feature provided by the PHY layer and literately comes at no cost at the link layer. Furthermore, the concept requires distinct roles assigned to the nodes, i.e., either receiving or transmitting the signatures, and can't be easily applied in fully decentralized scenarios.

Overall, the *Anatomy* is a very helpful scheme to understand and analyze the elements of a CR MAC protocol. It is also very helpful in understanding and specifying the dependencies and requirements as to what the different protocol parts need to be capable of.

The FLL architecture developed in this thesis builds upon these findings. However, it takes a more implementation-oriented view on the construction of the entire link layer, which, following our philosophy, consists of a set of decoupled management protocols that exploit the capabilities of a generic data transfer protocol.

3.1.5 Discussion

This section has reviewed existing efforts in creating architectures and frameworks for the development of flexible communication protocols that have been presented over the last twenty years.

This development was primarily driven by the desire to support varying application requirements and network conditions. Although many different solutions have been proposed, the core methodology behind them has largely been the same: a careful analysis of (common) functionality followed by the decomposition into modular building blocks and the subsequent reconstruction into new protocols. The actual way of doing so, however, often differs and seems to be, again, somewhat scenario-specific.

Table 3.1 shows a summary of the work reviewed in the transport/network protocol, in the WLAN/WSN, and the cellular domain. They are categorized according to the architecture model, the target protocol layer, the component granularity, the type of implementation, and whether they are extensible and adaptable. As can be seen from Table 3.1, the majority of frameworks is graph-based. Noteworthy exceptions are *MultiMAC* [72], using horizontally structured MAC layers, and GLL [99]. The research has exclusively dealt with the link, network, and transport layer. The granularity of components ranged from small, i.e., state machine elements, to medium, i.e., protocol fragments, to large, i.e., entire protocols. We could observe a general tendency towards fine-grained components the further the frameworks go down the protocol stack. With one exception, they were all implemented in software (SW). Understandably, all of them are extensible and more than half of them are also adaptable during runtime.

Table 3.2 shows a summary of frameworks for implementing flexible protocols in the CR and DSA domain. In addition to the attributes used above, we separate PHY and MAC implementation and also include spectrum mobility and link establishment into the discussion. Interestingly, all of the works follow a graph-based approach. They primarily focus on MAC and partly also on PHY layer issues. The modem part of the PHY layer is mostly implemented in software, either using GR or *Iris*, whereas the MAC is either implemented using *Click* or using the framework that has been used for PHY implementation. One noteworthy exception is the work done by Ansari *et al.* [103],

Table 3.1: *Summary of frameworks for implementing flexible protocols in the transport/network protocol domain, the WLAN/WSN domain, and in cellular networks.*

	Architecture model	Target (sub-)layer	Component granularity	Implementation	Extensibility	Adaptability
Hutchinson and Peterson [93]	stack-based	transport	large	SW	✓	-
Kohler *et al.* [94]	graph-based	network	medium	SW	✓	-
Kannan *et al.* [95]	graph-based	transport	medium	SW	✓	-
Bridges *et al.* [89]	graph-based	transport	small to medium	SW	✓	✓
Day [38]	stack-based	all	large	SW	✓	✓
Doerr *et al.* [72]	stack-based	MAC	large	SW	✓	✓
Bianchi *et al.* [70]	graph-based	MAC	small	HW	✓	✓
Polastre *et al.* [98]	graph-based	link/network	medium	SW	✓	✓
Klues *et al.* [97]	graph-based	link/network	medium	SW	✓	✓
Sachs [99]	stack-based	link	medium to large	n.c	✓	✓
Berlemann [107]	graph-based	link	medium	SW	✓	✓

Definitions:
- n.c. Not covered or discussed in appropriate detail
- SW Software
- HW Hardware

Table 3.2: *Summary of architectures and frameworks for implementing flexible protocols in the CR and SDR domain.*

	Target (sub-)layer	Component granularity	PHY implementation	Link/MAC implementation	Extensibility	Adaptability	Spectrum mobility	Link establishment
Dhar *et al.* [32]	MAC	small to medium	SW (GR)	SW (*Click*)	✓	✓	-	-
Mandke *et al.* [101]	MAC	small to medium	SW (GR)	SW (*Click*)	✓	✓	-	-
Yang *et al.* [102]	MAC+PHY	medium	SW (GR)	SW	✓	✓	n.c.	-
Ansari *et al.* [103]	MAC+PHY	small	HW	Hybrid	✓	✓	n.c.	-
O'Sullivan [24]	MAC+PHY	medium	SW (*Iris*)	Hybrid (*Iris*)	✓	✓	✓*	✓*

Definitions:
n.c. Not covered or discussed in appropriate detail
* Implemented as PHY technology

in which most parts of the radio are implemented on the FPGA. All reviewed frameworks allow reconfigurations at runtime but, in their current form, lack support for spectrum mobility and link establishment at the link layer.

With a single exception, CTP, this section has reviewed flexible protocol development frameworks but has ignored actual protocols. Therefore, the next section reviews a selected set of protocols for MAC, link establishment and spectrum mobility that have been proposed by the research community and are of interest to this dissertation.

3.2 Selected Link Layer Protocols

This section surveys a selected set of link layer protocols that have been proposed as either new protocols or extensions to existing protocols in order to solve a specific use case. We have again split the analysis into two categories: one covering MAC and spectrum mobility and the other covering link establishment protocols.

In the interest of space, we will only focus on work that is of relevance to the experimental evaluation presented in Chapter 5 of this thesis. For a comprehensive survey on existing link layer protocols for CR networks, we would like to refer the interested reader to Cormio and Chowdhury [110], De Domenico *et al.* [111], and O'Sullivan [24].

3.2.1 Medium Access Control and Spectrum Mobility Protocols

In this section, we discuss related work in the area of MAC and spectrum mobility protocols with respect to both algorithmic and architectural aspects. We first review two protocols presented in the context of overlay DSA environments. Second, we discuss two MAC protocols for fixed-spectrum networks that are capable to adapt their characteristics such that they better suit current network conditions.

SWITCH: A Multichannel MAC Protocol for Cognitive Radio Ad Hoc Networks [15]

SWITCH, proposed by Kalil [112, 15], is a multi-channel MAC protocol for CR ad-hoc networks based on IEEE 802.11. *SWITCH*, however, differs from IEEE 802.11 in several ways. First, *SWITCH* is a decentralized protocol that doesn't require a coordinating AP and works on multiple channels. Furthermore, it is designed to operate in DSA environments, which means it is capable to provide continuous communication, even if a PU occupies the channel currently in use by the protocol. It does so by providing a so-called backup channel (BC) that is negotiated among the nodes proactively, i.e., before the actual channel handover takes place. For this purpose, each node maintains two lists that are used to track channel activity. A so-called *neighbor channel list* for channels used by neighbor nodes and a *free channel list* for channels that are currently not in use. Both are updated through control messages sent over a CC shared by all nodes. In order to negotiate data and backup channel for communication, *SWITCH* exploits a two- or three-way-handshake procedure with request to send (RTS) and clear to send (CTS) messages. Both include additional fields that contain the proposed or selected data and backup channel.

From an algorithmic point of view, there are several disadvantages of *SWITCH*. First, RTS/CTS messages including data and backup channels are exchanged prior to each transmission, which may be unnecessary if network conditions didn't change. Second, the CC that is used for control message exchange may become a bottleneck in the network in certain traffic conditions.

From an architectural point of view, the monolithic design of *SWITCH*, i.e., the tight integration of spectrum mobility into the core protocol, makes it inflexible and difficult to reuse in other protocols.

C-MAC: A Cognitive MAC Protocol for Multi-Channel Wireless Networks [113]

C-MAC, proposed by Cordeiro and Challapali [113], is another decentralized multi-channel MAC for DSA environments. Unlike *SWITCH*, *C-MAC* is a time-slotted protocol, i.e., each channel is structured in the form of a super frame (SF). Each SF consists of two parts: a beacon period (BP) and a data transfer period (DTP). The BP is used for management functions of the protocol, such as adding and removing devices from a channel. This is done in the first two slots of the BP. From slot three onwards, each device belonging to a channel is required to broadcast a status beacon in its assigned slot. This beacon contains information about neighboring nodes, assigned beacon slots, and communication schedules.

One of the key features of *C-MAC* is its rendezvous channel (RC) concept. Among all available channels, the RC is a dynamically selected channel that is used for a variety of things such as group communication, neighbor discovery, synchronization, and load balancing. The protocol also supports spectrum mobility through backup channels that are also communicated among nodes through the BP.

Overall, *C-MAC* is a very advanced protocol which combines unique features such as time-slotted multi-channel access, group communication, and spectrum mobility. Beyond a conceptual description, however, the paper unfortunately only provides little details about the protocol and misses important aspects that would be needed, for example, for a detailed overhead analysis. This includes information about the DTP or the beacon frame structure, but also about the simulation and prototyping efforts, that would be required for a architectural analysis. It is noteworthy to say that parts of *C-MAC* are also part of the European ECMA-392 standard [114] for secondary access to TV whitespaces[1].

MAC Protocol Adaptation in Cognitive Radio Networks: An Experimental Study [115]

AMAC, presented by Huang *et al.* [115], is a protocol that supports MAC adaptation in order to support changing network scenarios. Specifically, it supports switching between CSMA and TDMA operation. The *AMAC* architecture is split into two parts: a global control plane that communicates on a dedicated radio channel (i.e., IEEE 802.11) and a data plane that is realized as a SDR. The actual *AMAC* protocol is implemented in the control plane and controls the MAC of the data plane. The decision and switching framework itself is based on "traffic prediction". In particular, future traffic patterns are predicted by measuring the average size of outgoing frames. If the average size is large, a node requests TDMA operation, if the size is small, CSMA operation is preferred.

From an algorithmic point of view, it is not clear how frame size can provide a meaningful switching indication. Transmission queue length could probably offer a more meaningful criterion, at least in non-saturated conditions.

From an architectural point of view, one drawback of *AMAC* is the assumption of an underlying control plane. As a result of this, it cannot be easily ported to single channel networks, using in-band signaling for example, without modifying the data plane.

Load Adaptive MAC: A Hybrid MAC Protocol for MIMO SDR MANETs [116]

LA-MAC, proposed by Hu *et al.* [116], is a further MAC protocol that tries to combine the advantages of random-access, i.e., CSMA, and time-slotted protocols. *LA-MAC* has two operation modes. During initialization, the protocol runs in CSMA mode only. Its main task is collecting neighbor information, synchronizing time, and assigning time slots. During normal operation, *LA-MAC* differentiates between two operation states: low contention (LC) and high contention (HC). While in LC state, *LA-MAC* behaves like standard CSMA. In HC state, it exposes its time-slotted characteristics. Switching between LC and HC states is done by each node individually based on notification messages received from neighbor nodes. The paper provides some details on the implementation of *LA-MAC* using GR, which inherently yields a component-oriented design. However, the description indicates that a rather monolithic approach has been taken for both the integration of the TDMA functionality into the existing CSMA-MAC and the implementation of the switching functionality. From an architectural point of view, this again prohibits MAC reuse.

[1] C. Cordeiro, personal communication, October 2014.

Table 3.3: *Summary of rendezvous algorithms for centralized and decentralized networks without control channel.*

Algorithm	Scenario	Channel model	Guaranteed rendezvous
Random [52]	both	both	no
MC [52]	both	symmetric	no
MMC [52]	both	both	no
JS [53]	both	both	yes
EX [56]	centralized	both	yes

3.2.2 Link Establishment Algorithms and Protocols

We will now briefly describe a number of rendezvous algorithms that are later employed to showcase the link establishment capabilities of the FLL architecture. Those algorithms include solutions for both centralized and decentralized networks. However, none of them requires a CC to function properly. Table 3.3 briefly summarizes the selected algorithms according to the supported network scenario, their channel mode, and whether guaranteed rendezvous can be provided.

Random Rendezvous [52]

The random rendezvous algorithm is a basic algorithm that works under any network scenario and channel model but cannot guarantee successful rendezvous [52]. In this algorithm, each node i hops over the set of available channels C_i until rendezvous occurs. In each slot, i selects a new channel c from C_i with probability $p = \frac{1}{|C_i|}$. A link between two nodes can be successfully established if two nodes select the same channel in the same slot.

Modular Clock (MC) and Modified Modular Clock (MMC) [52]

Modular Clock (MC), proposed by Theis *et al.* [52], is a rendezvous algorithm that uses prime number modular arithmetic for creating CH sequences.

The CH sequence generation starts by randomly selecting a channel index j from the set of available channels C, with $|C| = M$. In each time slot t, the next index j_{t+1} is generated by advancing the previous index with a certain step-length r, also chosen randomly, between $[0, P]$. P is the smallest prime number greater than or equal to M. Anytime the algorithm outputs a channel index greater than M, the modulo operation is applied to remap the selected channel. Hence, j_{t+1} may be calculated as follows:

$$j_{t+1} = (j_t + r) \, mod \, P \, . \tag{3.1}$$

The algorithm is shown to be more efficient than the random algorithm for more than three channels under the symmetric model, if both users select the same prime number P but different rates r [52].

In the asymmetric case, rendezvous can be achieved in P time slots, if both users select different prime numbers. It should be noted, however, that MC cannot guarantee rendezvous because it cannot be guaranteed that two users always select different rates and prime numbers [55].

To avoid the case where two users select the same prime numbers, the selection process was modified in an extended version of MC, called *Modified Modular Clock* (MMC). In MMC, P is chosen from the set of primes in $[M, 2M]$. Furthermore, the channel remapping function was modified. In MMC, the new channel is chosen randomly if the generated channel index exceeds M.

Jump and Stay (JS) [53]

Jump and Stay JS, proposed by Liu *et al.* [53], is a round-based rendezvous algorithm that generates CH sequences with a length of $3P$ slots in each round, where P is the smallest prime number greater than M. In each round, a node jumps over the set of available channels for $2P$ slots (jump period) and stays on a specific channel for the remaining P time slots of the round (stay period). To generate the patterns in each round, each node has to randomly select a step-length value r between $[1, M]$ and an index i between $[1, P]$. In the jump period, the node starts with index i and keeps hopping over C with step-length r for $2P$ slots. During the stay period, each node stays on channel r for P slots.

As the algorithm may run for more than one round, i.e., more than $3P$ slots, step-length r and index i need to be modified in each round in order to generate different jump and stay patterns. Therefore, r is switched to the next number in $[1, M]$ (with rewrap, i.e., applying modulo M) every $3P$ slots. Furthermore, i is switched to the next number in $[1, P]$ (applying modulo P) every M rounds, i.e., after $3MP$ slots. Anytime the algorithm outputs a channel index greater than M, the modulo operation is applied to remap the selected channel.

During the algorithm analysis, the authors of JS proved that it can guarantee rendezvous. Specifically, in the symmetric case, JS requires at most $3P$ time slots to achieve rendezvous. In the asymmetric case, it requires at most $3MP(P-G)+3P$ time slots.

Among the reviewed algorithms for decentralized networks, JS is the most powerful approach because it supports both symmetric and asymmetric channels, and is able guarantee rendezvous. We therefore select JS as the candidate solution for decentralized networks and further assess its performance in Chapter 5.

Exhaustive Search [56]

For establishing links in centralized networks without using a statically defined CC, Kondareddy *et al.* [56] have proposed an algorithm that performs an exhaustive search (EX) over the set of available channels. The key idea behind this strategy is to exploit the heterogeneity of nodes in the network, i.e., their different roles. Consider a typical infrastructure-based scenario, for example. In such a scenario, both the leader, e.g., the base station, and the follower, e.g., the mobile users, hop over the set of available channels. The difference between both is their hopping rate. While the leader node picks a new channel in every time slot, followers only update their operating channel every M_i slots, where M_i is the number of available channels to node i with which rendezvous should take place, in our case the leader node. As the exact number of channels available to a node is generally unknown, M can be taken as an upper bound.

Assuming two nodes C_1 and C_2, this algorithm guarantees to achieve link rendezvous between both nodes in at most M^2 time slots, or $M_1 \cdot M_2$ if the exact number of channels is known, assuming they have at least one channel in common.

Leader and follower hop over their set of available channels starting with the lowest channel. The leader repeats this procedure until link rendezvous occurs. In principle, the follower employs the same procedure with the main difference, that it only tunes to a new channel every M time slots. It remains in one channel for M slots in order to allow the follower to meet the leader in one of the commonly available channels.

Because EX is the only algorithm that is applicable in the centralized scenario only, it is selected as the candidate solution for centralized networks. Therefore, we will further asses its performance and compare it against random rendezvous and JS in Chapter 5.

3.2.3 Discussion

In this section, we have reviewed a selected set of link layer algorithms and protocols targeted for FWCS. Table 3.4 compares them according to their functionality, the means of evaluation, and their architecture. While algorithms for link establishment have solely been studied as standalone entities, spectrum mobility has mostly been considered as an extension to an existing MAC protocol.

Due to the nature of the problem, rendezvous algorithms have been evaluated mostly through mathematical analysis and simulation. One notable exception is the work carried out by Silvius *et al.* [117]. In Chapter 5, we will extend their work by integrating link establishment capabilities into our FLL architecture. Furthermore, we will study the performance of three rendezvous algorithms through simulations.

In contrast to rendezvous, MAC and spectrum mobility protocols have also been studied practically. However, flexibility and functional reuse have not been considered in the reviewed protocols. These aspects will be addressed in the next chapters.

Table 3.4: *Summary of reviewed link layer protocols.*

	MAC/data transfer	Link establishment	Spectrum mobility	Means of evaluation	Separation of concerns
SWITCH [15]	✓	-	✓	A, S	-
C-MAC [113]	✓	✓	✓	A, S, I	-
Huang *et al.* [115]	✓	-	-	I	-
Hu *et al.* [116]	✓	-	-	A, I	-
Random [52]	-	✓	-	A, S	n.a.
EX [56]	-	✓	-	A, S	n.a.
MC [52]	-	✓	-	A, S	n.a.
MMC [52]	-	✓	-	A, S	n.a.
JS [53]	-	✓	-	A, S	n.a.

Definitions:

A	Analysis
S	Simulation
I	Implementation
n.a.	not applicable

3.3 Summary

The dream of an always connected society with billions of communicating devices and new application areas poses significant challenges on following generations of wireless communication systems. The diversification of requirements, applications, and operating environments demands FWCS to be as flexible and adaptive as possible. Naturally, this also leads to an increase in functional complexity. In order to keep this complexity manageable, it is equally important to also rethink their design and implementation process.

In the past, the desire to support varying application requirements and network conditions has led to the development of various frameworks for developing flexible communication protocols, most of which are based upon modular, recomposable building blocks. However, the underlaying architectural model, target protocol layer, and the granularity of modules have differed across them. Sadly, even forward thinking strategies have mostly failed in providing combined solutions beyond simple MAC, jointly addressing link establishment and spectrum mobility issues.

The vast majority of research in the CR/DSA domain has simply ignored architectural aspects of protocols. Possibly because their authors never attempted to implement and evaluate them in real-world systems. But even if they have been implemented, for example in simulation frameworks, the protocol description allows to assume a tight integration of new features into the protocol core, thus yielding a monolithic architecture. However, by mixing data transfer and management functions of a protocol, one also limits the possibility of reusing common functions in multiple protocols.

Furthermore, mixing of functions also narrows the flexibility of combining existing blocks to entirely new protocols. From a practical perspective, monolithic designs are also more complex and difficult to implement and maintain.

Therefore, the goal of this dissertation is to expand the current scope of flexible development frameworks and implementation strategies for future communication protocols. The analysis has shown that a protocol architecture has to be created that provides proper and coherent methods for functional separation. In the next chapter, we develop a basic link layer framework that provides such a coherent set of methods for designing and implementing protocols that are maintainable, extensible, and adaptable. The underlying concepts of the architecture are based on two fundamental principles of system engineering which will be explained first.

4 Flexible Link Layer Architecture

The previous chapters have introduced FWCS and highlighted the challenges they have to overcome. These challenges included the massive growth in traffic volume and number of connected devices, but also the diversification of applications and operating environments. Among a number of technology components that will be part of FWCS, we have then presented DSA systems as one possible approach to mitigate the effects of inefficiently assigned radio spectrum in today's networks.

Due to the dynamics of the underlaying PHY layer, the link layer of DSA systems faces significant challenges when it comes to establishing links among nodes and maintaining those during communication. Unfortunately, there is no universal solution to overcome them, primarily because they are highly application- and scenario-specific. We therefore argued that flexibility will be of paramount importance throughout the entire life-cycle of FWCS, including development and deployment. In our opinion, existing approaches towards flexible link layer protocols have failed in providing satisfying solutions.

This chapter presents the *Flexible Link Layer* (FLL) architecture, our approach to address some of the key challenges. This architecture has been first introduced at the *8th International Conference on Cognitive Radio Oriented Wireless Networks (CROWNCOM)* in 2013 [7]. Later in 2014, an evolved and more generic version of the architecture has been presented at the *9th ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH)* [5]. In the following, we will describe the entire architecture, the underlying concepts, and its core components in detail. We will begin by restating the research problem and highlighting the core ideas behind the FLL architecture.

4.1 Problem Analysis and Solution Idea

In the previous chapters of this dissertation, we have learned that the functionality of the link layer of future DSA system is not fundamentally different to traditional systems. Besides the basic functions, however, it has to cater for a number of management-related aspects due to the increased variability of underlying PHY layer. As has been shown in Section 3.2, the usual

approach to address them has been to integrate the required management functions into an existing protocol. Unfortunately, this usually yields to a monolithic design that is more complex and neither reusable nor flexible. But why is that the case?

Problem Analysis

We believe one of the core problems of many protocols is the *mix of data transfer and management functions* inside a single entity. From an engineering point of view, this integration, as an evolutionary process, is understandable because both functions are closely related to each other. Messerschmitt put it aptly when he said that tailored designs provide "the shortest route (in terms of time and cost) to realization" [84].

There are good reasons, however, for keeping data transfer and management function separated. First, if a management function becomes part of a data transfer protocol, the management function increases the complexity of the data transfer protocol. This increase in complexity not only results in more code needed for the implementation of the protocol, it may also increase the protocol overhead during communication. Second, should the management function not be required, the data transfer protocol cannot be easily used without it. The reason for this is primarily the domain-specific management part that cannot be separated from the generic protocol. This also hinders reusing generic components which provide fundamental functions. Such functions shouldn't be replicated multiple times within a layer. Furthermore, they shouldn't be spread across a single layer. Not being able to separate functions from each other also makes evolution more difficult and causes difficulties with legacy components [69].

We believe a second problem in many link layer protocols is the *mix of media-independent and media-dependent functions*. This fact can also be explained through a simplified engineering process. However, by keeping both aspects separated, chances are much higher that generic functions can be reused, and only small parts need to be changed in future systems. For example, if a CSMA-MAC was to be replaced by a TDMA-MAC, the error control capabilities of a separate component could remain in place

But how can we create an architecture that avoids the above stated mix of responsibilities? Towards finding an answer to this question, the next section will introduce the key design concepts of the FLL architecture.

Solution Idea

After having highlighted the main issues of current protocols, we can now formulate the core design philosophies of the FLL architecture which can be summarized as follows:

- **Modeling management functions as link layer applications:** Instead of adding management-related aspects to a data transfer protocol, they should be kept separate. Specifically, we propose to understand and model management functions as "applications" inside the link layer architecture. This allows the data transfer protocol to concentrate on the exchange of information instead of dealing with management-related aspects, as intended by the SoC concept. Link layer applications, like ordinary user applications on top of the layer, use the highly specialized data transfer protocol to exchange information among each

other. Ideally, this allows to exchange both the data transfer and the management protocol, without modifying one or the other.

- **Separation of media-independent and media-dependent functions:** This separation refers to solving the issue of combined responsibilities with respect to the core link layer functions and MAC. Instead of addressing core link layer concerns such as error control inside a media-dependent protocol, following the SoC concept, they should be kept at two distinct places. However, both should be connected through an interface that doesn't restrict either of the two.

Admittedly, the idea of separating media-independent and media-dependent functions is nothing entirely new. In fact, numerous communication technologies employ the concept of separating error control and MAC into distinct components (or sub-layers), for example cellular systems like LTE [118]. However, taking IEEE 802.11 as a prominent example that forms the basis for a great many protocols, we will show later in this and the next chapter how this principle is often violated in practice.

Design Choices

Having introduced the core idea of the FLL architecture, a number of questions related to an actual realization come to mind. The next section discusses them in detail. Before that, we will briefly highlight some of the key concepts of the FLL architecture and describe how those help addressing these questions.

The first unanswered question is that of component granularity. Choosing the granularity of components in a system is a trade-off between a large number of relatively small components (fine grain) and a small number of relatively complex components (coarse grain). The review of existing approaches in Chapter 3 has explained the dilemma. To avoid having to make a hard choice, we propose to use *hierarchical componentization* as "a compromise between generality and specificity" [84]. Hierarchical componentization allows components themselves to be decomposed into smaller components in order to restrict the amount of interacting components at each level. From a functional perspective, components on the top level of the hierarchy embrace rather system-specific aspects, such as providing a "communication service". Components at lower levels of the hierarchy are less system-specific but more technology-specific. For example, this allows to implement a specific MAC protocol with fine grain components at state-machine level (e.g., following the approach proposed by Ansari *et al.* [103] or Bianchi *et al.* [70]). On the other hand, media-independent functions could be implemented in a different protocol using larger components, as has been pursued by Bridges *et al.* [89]. Even extremely large components at protocol level are imaginable. For example, one could imagine implementing an entire link establishment protocol within a single component.

The second question is that of deciding how to model dynamic behavior between components. The FLL architecture accommodates for this by introducing a link layer controller. The primary functions of the controller are mediating and coordinating the interactions among the set of link layer components. These interactions are modeled through an FSM that implements the *business logic* of a certain link layer configuration. For example, what needs to be done if an active PU has been detected?

The third question that arises in component-based systems is that of defining interfaces through which components can communicate with each other. Within in the FLL architecture, control information between components is exchanged via events that are coordinated and dispatched through the link layer controller. In contrast, data that is communicated from one node to another is always transmitted through the GDTP. Note that this also applies to data that is exchanged between instances of a specific link layer application running on different nodes, even though this data, from an overall link layer point of view, may be control data.

In the next section, we will describe the entire FLL architecture and its building blocks more detailed.

4.2 Architecture Synthesis and Design

In this section, we describe the main components of the FLL architecture. However, we will not be concerned with specific algorithms or protocols but rather with the type of service that is provided. Specific algorithms will be defined and their implementation using SDRs will be discussed in Chapter 5. The FLL architecture comprises four core components:

- **Generic Data Transfer Protocol:** The generic data transfer protocol (GDTP) is the most important component of the FLL architecture. The GDTP provides the basic link layer functions including error and flow control.
- **Medium Access Control:** The medium access control component provides functions that depend on the underlying communication medium.
- **Link Layer Applications:** Link layer applications (LLAs) implement management-related concerns of a link layer protocol.
- **Link Layer Controller:** The link layer controller is the component that connects the individual components with one another and mediates their interactions.

Figure 4.1 shows the core components including their data and control connections. Note the vertical separation between management and data transfer related functions as well as the horizontal separation between media-independent and media-dependent functions within the data transfer related part.

Note that the FLL architecture is a framework for constructing link layer protocols using components. It is, however, not a specific protocol. Distinct link layer configurations may be created for various use cases by combining LLAs, controllers, or MACs together with a GDTP implementation. Thus, a link layer configuration can be understood as a distinct set of protocols and protocol parameters that together provide the required functionality and behavior. Each of the core components will be described in detail in the remainder of this section.

4.2.1 Generic Data Transfer Protocol

In the broadest terms, the responsibility of the GDTP is to provide basic data transfer services to its clients. GDTP clients may be ordinary user applications on top of the FLL as well as other FLL components that exploit its data transfer facilities.

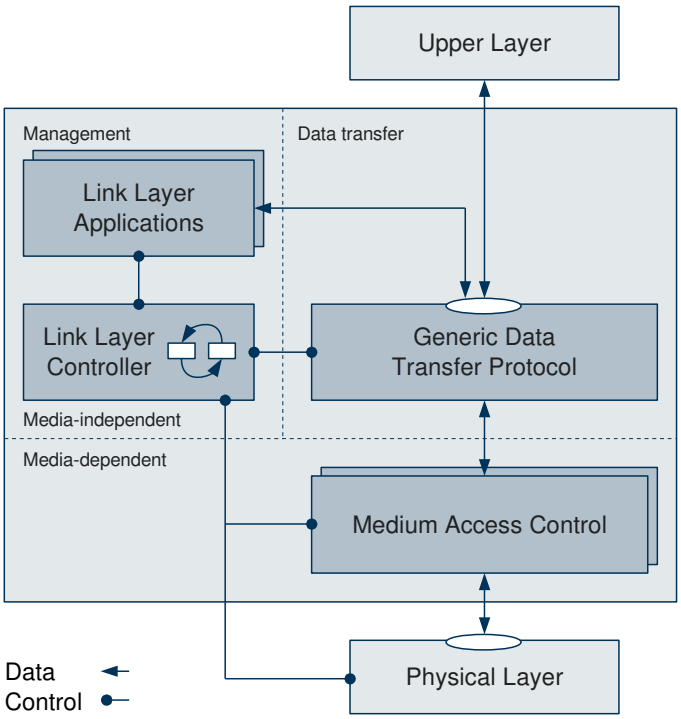


Figure 4.1: *High-level design of the FLL architecture comprising its four major components.*

Clients of the GDTP may include:

- higher layer protocols, such as a routing or transport protocol,
- ordinary user applications also implemented in the layer above the FLL, such as a video or sensor data streaming application, or
- LLAs, such as a spectrum mobility protocol, which are part of the FLL architecture themselves.

To support heterogeneous communication requirements, the provided level of service is not fixed but can be controlled by each client individually. For example, each client may select between unreliable service, e.g., for video streaming, or reliable data transfer, e.g., for file transfer or sensor data dissemination.

In order to provide its services, the GDTP includes a number of protocol mechanisms, such as framing, error control, lost and duplicate detection, multiplexing, and ordering. However, the GDTP doesn't include any media-dependent functions and also doesn't support multi-hop communication. In other words, it misses a routing and forwarding service.

For specifying the offered services, the GDTP provides an interface that allows to explicitly define certain communication characteristics on a per-flow basis. Based on the specific requirements of each user, different policies are applied to the provided mechanisms in order to fulfill the desired goals. For distinguishing between multiple clients between the same source and destination, the GDTP internally uses flows. Note that we view a flow as a "sequence of packets

sent from a particular source to a particular unicast, anycast, or multicast destination" [119], but not necessarily as a connection-oriented communication session, such as TCP. Figure 4.2 shows the details of the GDTP and how it is embedded in the FLL architecture when viewed through a magnifying glass.

In the context of this thesis, the GDTP shouldn't be viewed as a replacement for ordinary transport protocols, such as UDP or TCP. In fact, its main purpose is to provide a configurable, multi-user, media-independent data transfer protocol on top of a potentially unreliable, single channel lower layer.

In the remaining part of this section, we will briefly describe the components that are most relevant to this thesis and mention possible mechanisms and policies that the protocol could implement. This discussion is loosely based on the set of protocol elements described by Day [38]. However, we entirely omit security related mechanisms, such as authentication and access control.

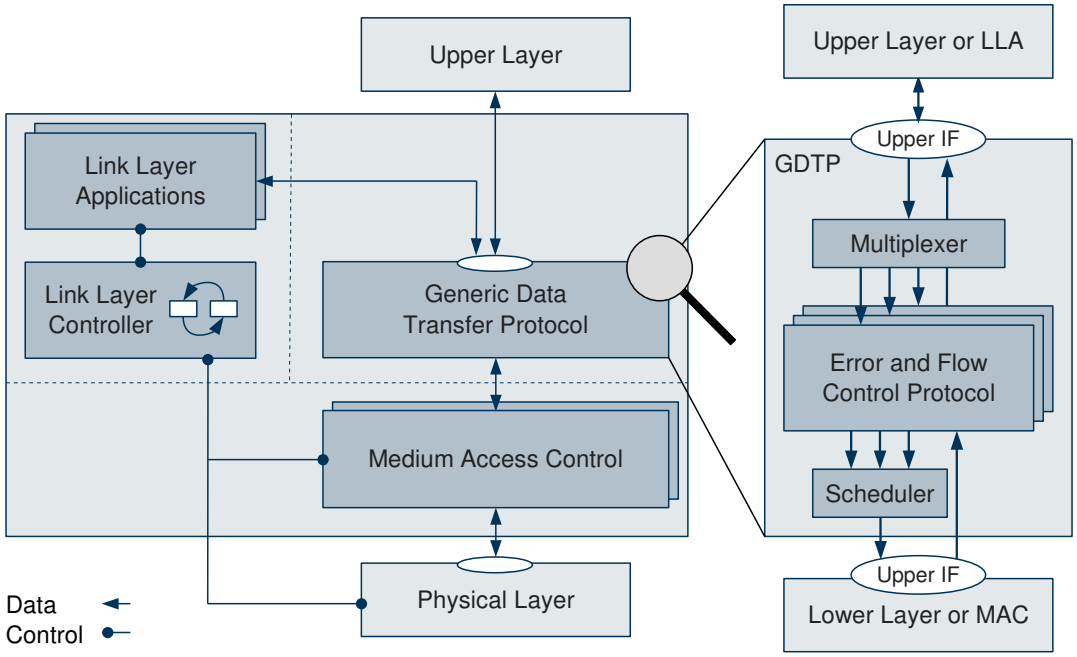


Figure 4.2: *The basic architecture of the generic data transfer protocol comprising an upper interface, a multiplexer, an error and flow control protocol instance (per flow) as well as a scheduler that coordinates multiple contending flows.*

4.2.1.1 Protocol Interfaces

Protocol layers require interfaces that allow clients on top or below the layer to interact and exchange data with the layer. In case of the GDTP, a client on top of the protocol could be an ordinary user application, as has been discussed above. By clients below the protocol, we mean a MAC or PHY layer, for example, that uses the lower interface to deliver incoming frames. Depending on the actual implementation, however, the GDTP may not have a lower interface.

Instead, it is a client itself that uses the interface provided *by* the lower layer. We discuss possible design alternatives in the remaining part of the paragraph.

Upper Interface

The functions of the GDTP are provided to its clients through an application programming interface (API). The API allows to create (or allocate) a new flow, to send and receive data on a given flow, to read and write the characteristics of a flow, and to close a flow, i.e., to deallocate it. The upper interface provides a technology-agnostic means for specifying the desired end-to-end data flow characteristics. Furthermore, their actual state may be determined through the interface. The core properties that may be set and read on a per-flow basis can be summarized as follows:

- **Throughput:** Data rate (in kbit/s) provided to the user.
- **Delay:** Maximum communication delay (in ms) between transmitter and receiver.
- **Reliability:** Maximum frame error rate (FER) between transmitter and receiver (measured in a certain time interval).
- **Priority:** Relative importance of a flow (represented as a sortable number).

Furthermore, one may specify other flow properties that influence the operation of the layer. Those properties may include ordering of frames, concatenation, segmentation, and padding of PDUs. Additionally, the interface may expose further information on the connection state (e.g., connected or unconnected) that may be used by higher layer protocols. For example, a mobility management protocol may use such information [99].

This interface serves as an, admittedly, idealistic generic layer abstraction, as can also be observed from the direct mapping of the link layer interface on the GDTP interface in Figure 4.1. This is also because not all of the specified metrics can be enforced directly and alone through the GDTP itself. For example, effective end-to-end error rate may be influenced by a number of mechanisms at the link and PHY layer, such as the employed ARQ or FEC scheme. However, this may be dependent on the characteristics offered by the lower layer and may not be visible to the GDTP. Therefore, the process which decides upon the selected policies and when they are changed resides outside of the GDTP itself. For example, this could be realized inside an optimization module that jointly modifies link and PHY layer parameters. However, the implementation of such a module is out of scope of this thesis.

An abstract interface of the GDTP, based on the generic API described by Day [38, p. 194], may look as follows:

- <result> ← Allocate(<flow-id>, <destination>, <application-name>, <properties>)
- <result> ← Send(<flow-id>, <data>)
- <result> ← Receive(<flow-id>, <buffer>)
- <result> ← Modify(<flow-id>, <properties>)
- <result> ← Deallocate(<flow-id>)

A client of the GDTP creates a new flow by invoking the Allocate() call and defining the destination, the name or ID of the application, and the desired communication properties. If a connection could be established, the Allocate() call returns the ID of the created flow. If something went wrong during allocation, the result may include an error message. After a flow has been created successfully, the flow ID can be used to send and receive data. During communication, the properties of a flow may be changed through the Modify() primitive. At the end of a communication session, a flow may be closed by invoking the Deallocate() API call.

For the upper interface, such a dual "push-pull mechanism" in which the client of the GDTP initiates any interaction is common practice. However, this might not be the case for the lower interface.

Lower Interface

Understandably, there also needs to be a mechanism that allows to deliver incoming PDUs from a lower layer as well as to transmit outgoing PDUs on that lower layer.

However, unlike the upper interface, the lower interface may follow a push, pull, or a push-pull strategy, depending on the actual implementation. For example, the communication system could be designed such that the GDTP pulls incoming frames from the underlaying PHY layer. In this case, the GDTP wouldn't need to provide an own lower interface. In interrupt-driven systems, however, it is common practice that the lower layer informs the upper layer about incoming frames. In such a case, the GDTP needs to provide a proper lower interface.

In our prototype implementation presented in Chapter 5, we have also employed the push-pull mechanism. In other words, the GDTP itself uses the upper interface of a lower layer component and pulls incoming frames off the interface and pushes outgoing frames through the interface.

For proper handling of timers, the lower layer interface may also indicate when a queued PDU has actually been transmitted. It may also allow to delete an outgoing PDU from the transmission queue. For example, this could be useful to delete pending retransmissions if a delayed acknowledgement arrives. Furthermore, the interface may indicate the amount of data the lower layer can transmit as well as statistical information about lost or dropped frames, or other quality estimates. This information may also be used by the above mentioned optimization module.

It is worth mentioning that the GDTP expects frames received from the lower layer to be error free. In other words, there needs to be an error detection mechanism inside the lower layer.

Modeling Multi-Channel Systems

When talking about multi-channel systems, a further question to answer is that of how to model those from the perspective of the data transfer protocol. Clearly, if the number of physical communication interfaces, e.g., PHY transceivers, is smaller than the number of orthogonal channels, there are channels that are "unconnected" and, therefore, unusable from the viewpoint of the GDTP. In this case, the differentiation between orthogonal and non-orthogonal is important. This is because orthogonal link configurations usually break the link if they change. For example, if the transmitter changes its center frequency, the receiver also has to change its center frequency to still be able to receive the transmission. Non-orthogonal configurations do not necessarily break the link if they change. They usually contain continuous parameters, such as transmit power. For example, if a transmitter reduces its transmit power with each frame it transmits by 3 dB, it may

still be possible to send several frames before it eventually reaches a certain power level at which the receiver can no longer decode the frame. Therefore, orthogonal reconfigurations always have a direct impact on the network, because they effectively cause a topology change, i.e., change the unconnected/connected links. In contrast, non-orthogonal reconfigurations may or may not cause such a change.

Two strategies are conceivable for representing different channels inside the GDTP. One is to include unconnected/unusable channels into the GDTP, i.e., to model each possible channel as one point of attachment (PoA). This requires a mechanism to control the PoA and its actual configuration for each flow. The second option is to have exactly one PoA per physical interface that is reconfigured by another component, for example the link layer controller. The advantage of this approach is that from the perspective of the GDTP, the PoA doesn't change, during spectrum mobility for example. In fact, this strategy has been adopted in the GDTP prototype implementation (see Section 5.3.1).

4.2.1.2 Initial State Synchronization and Policy Selection

This mechanism initializes the shared state between two protocol instances prior to any information exchange. This includes the selection of policies, e.g., the selected level of reliability, but also the creation of local bindings, and the initialization of sequence numbers. Four basic forms for achieving this synchronization can be found [38]:

- implicit shared state creation without exchanging control frames,
- two-way handshake through the exchange of request-response frames,
- three-way handshake, and
- timer-based mechanisms.

In addition to the initial synchronization mechanism, adaptive protocols may also require a mechanism that allows to change the value of a certain feature during communication, e.g., to increase the size of the transfer window. This is sometimes also known as "feature negotiation" [120]. From an architectural point of view, such a mechanism could be either implemented as a separate LLA on top of the GDTP or as a mechanism of the GDTP itself. The GDTP prototype employs a timer-based mechanism for initial state synchronization but misses a generic mechanism for runtime feature negotiation.

4.2.1.3 Addressing

Addressing is an important aspect in computer networks, especially in broadcasting systems without directly connected communication endpoints. Therefore, if a protocol data unit (PDU) is transmitted over a shared commutation medium, it carries two identifiers (or addresses) in its protocol control information (PCI) to unambiguously identify its origin and destination. Based on these addresses, the protocol machine can decide whether it accepts or discards an incoming frame. Depending on the configuration of the protocol, a GDTP may be assigned multiple addresses to listen to. For example, if a node has multiple physical interfaces or supports multicast

or broadcast traffic, it may listen for more than one destination address. Similarly, a GDTP may also use more than one address to tag outgoing frames.

Although addresses are required by the protocol in order to function properly, addressing itself is not part of the GDTP. This includes assigning addresses or identifiers to applications and nodes as well as performing address resolution, i.e., translating names to unique addresses in the current scope of communication.

If running on top of a layer that provides direct point-to-point connectivity, the GDTP may even run without needing to worry about addressing at all. Although possible, this mode of operation is not further analyzed in this thesis.

4.2.1.4 Flow Identification and Multiplexing

In order to distinguish multiple flows between the same source and destination, a flow identification field is used. This identifier is needed to deliver the payload of incoming PDUs to the right LLA or upper layer application. Upon reception of a data PDU that can't be assigned to an existing flow, a new flow with a default configuration and a unique identifier can be created automatically. This identifier needs to be unambiguous between the protocol instances. Similarly, each flow is assigned a unique lower layer port that is used to deliver outgoing frames to the right lower layer instance.

4.2.1.5 Error and Flow Control Protocol

As already mentioned above, the data transfer operation between two communicating endpoints is handled through an error and flow control protocol (EFCP). Depending on the specific requirements of the client, the GDTP should be able to apply different policies in both directions.

Therefore, the EFCP is required to differentiate between ingoing and outgoing flows. An outgoing flow on a transmitter, each having its own transmission queue, is processed as an incoming flow at the receiver. Therefore, a single node maintains multiple instances of the same EFCP, each serving a single flow.

The EFCP uses sequence numbers to ensure error control, flow control, and ordering of frames as desired. Each PDU of the protocol includes a sequence number field that the receiver can use to tell the original sender whether the PDU has been successfully received or not. This is called an acknowledgment (ACK). If the transmitter of a PDU does not receive an ACK within a specified amount of time, it assumes that either the PDU itself or the ACK got lost and retransmits it. This mechanism is referred to as automatic repeat request (ARQ).

The most simple ARQ mechanism is the stop-and-wait mechanism that sends exactly one data PDU at a time and waits for an ACK before the next data PDU is sent. However, this protocol has a relatively large overhead because every data chunk is acknowledged individually. This may result in low end-to-end throughput, especially in environments with a large bandwidth-delay product (BDP). In such a case, more sophisticated mechanisms could be used that employ a sliding-window mechanism to allow multiple frames to be transmitted before the sender expects an ACK for them. Furthermore, negative acknowledgments could be used to inform the sender about lost frames. For more details on ARQ protocols, we would like to refer the interested reader to Tanenbaum [36].

In summary, the EFCP is the component that ensures the desired level of reliability between two communicating nodes and makes sure that the receiver is not overloaded.

4.2.1.6 Scheduling

Similar to the scheduler of an operating system that coordinates the access of multiple processes to a single resource, the GDTP also requires a scheduler component. The purpose of this scheduler is to organize the access of multiple contending EFCP protocol instances to the same lower layer, e.g., the PHY layer. This component is required to maintain a list of all active flows. It's main purpose is to determine the order in which they can access the shared resource. Usually, there is exactly one scheduler per lower layer instance, i.e., one scheduler per radio transceiver in a wireless communication system.

Note that this scheduler is different to MAC, which will be discussed shortly. This scheduler manages contending flows trying to access the radio transceiver on the *same* node, rather than multiple contending nodes trying to access the actual physical medium. Typical frame schedulers are first-in first-out (FIFO), round-robin (RR), and priority-based.

To give an example, consider three applications that all have data to transmit, see Figure 4.3. Each of the three flows is assigned a different traffic class, i.e., a different priority. Based on the priority of all flows that are ready, the scheduler decides which one to activate.

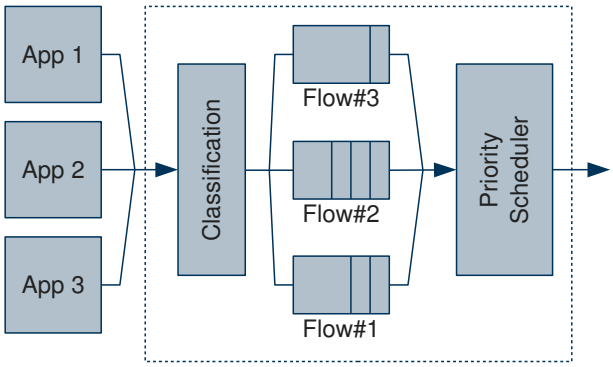


Figure 4.3: *Example of a priority-based scheduler. Incoming frames are classified and assigned to a flow. Among all flows that are ready, a priority scheduler selects the one with the highest priority for transmission.*

The scheduler is also among the components responsible for the QoS of the system. Therefore, the GDTP implementation should provide an interface for switching between different schedulers, if so desired. However, the process which decides upon the selected scheduler is again out of scope of the GDTP.

In the context of the FLL architecture, it is of paramount importance that the employed scheduler is able to fulfill the communication requirements of all LLAs. For example, a LLA may have certain bandwidth requirements or needs to transfer periodic messages. Understandably, this traffic should be handled with care. In most cases, the traffic of LLAs will have a higher priority than ordinary user traffic.

4.2.1.7 Protocal Data Unit Format and Serialization

As any other communication protocol, the GDTP needs to define a PDU format that is used to transfer information between two protocol instances. A GDTP PDU consists of a header that contains the PCI and an optional service data unit (SDU) part that carries the payload of the upper layer protocol. SDUs are transmitted transparently between two GDTP protocol instances. The PCI at least includes the source and destination address of the PDU, the PDU type, the flow it is associated with, and a sequence number. The GDTP at least requires one data PDU, but may define more than that to implement its features. For a discussion on how many PDUs a protocol should have, including a historical perspective on the issue, we again refer the interested reader to Day [38, p. 76].

Before PDUs are sent from one protocol instance to another, they need to be serialized (or encoded) into an array of bytes that is given to the lower layer. On the receiving node, this array of bytes is provided by the lower layer and needs to be deserialized (or decoded) into a valid PDU.

4.2.1.8 Concatenation and Segmentation of Frames

Every frame that is transmitted over the GDTP has a fixed overhead associated with it. If only a short payload is to be transmitted, the overhead of a frame can easily exceed the actual amount of information that it carries. To reduce the relative overhead per transmitted frame, it should be possible to concatenate multiple frames into a single frame[1].

Moreover, a GDTP may also allow to segment frames from a higher layer protocol into multiple chunks if they are larger than the maximum transfer unit (MTU) of the lower layer protocol. Figure 4.4 illustrates the data flow through the protocol layers.

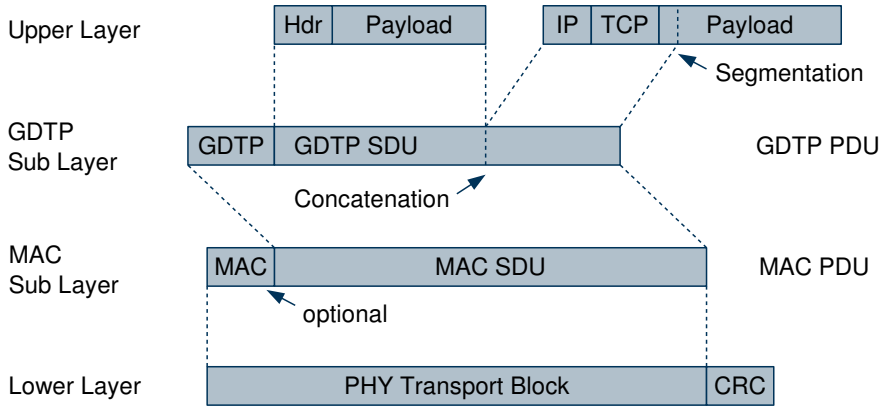


Figure 4.4: *Illustration of the data flow between upper and lower protocol layers (derived from [118]).*

[1] This feature is also known as frame aggregation within the IEEE 802.11e and IEEE 802.11n wireless LAN standards.

4.2.2 Medium Access Control

The MAC component of the FLL architecture views MAC in its purest sense, i.e., as a function required to coordinate multiple network devices while accessing a shared physical transmission medium. As a consequence, the MAC component doesn't provide any error control mechanisms. Note that this is in contrast to IEEE 802.11, for example, in which MAC and error control are tightly intertwined, even though "sublayering" also exists in IEEE 802.11.

As discussed in Section 2.3, the division of the shared medium can either happen in time, frequency, code, or space. Needless to say, the MAC component should adhere to the employed scheme.

However, it should also be noted that the separation between GDTP and MAC sometimes comes at an additional cost. For example, in order to schedule retransmissions at the correct instant of time, there needs to be a feedback mechanism that tells the GDTP when a frame has actually been transmitted at MAC or PHY level. Moreover, additional control information may be required to treat data and control PDUs differently, if this should be required. However, we believe these disadvantages can be compensated for in a careful protocol design.

In the next chapter, the benefits of the separation will be shown. Specifically, we demonstrate how the FLL allows to exchange the MAC scheme without having to sacrifice reliability.

Isn't IEEE 802.11 already divided into sublayers?

The IEEE 802 standard [121] defines a devision of the link layer into two sublayers: the logical link control (LLC) sublayer as the upper part and the MAC sublayer as the lower part. The main motivation behind this separation was to have a generic, media-independent part and a media-dependent part, tailored to a particular medium being used, such as Ethernet or token ring.
However, when IEEE 802 was defined, it was assumed that the underlying PHY layer is almost error free. Therefore, the LLC sublayer only includes features such as multiplexing, but has no support for error control, i.e., it has no ARQ mechanism whatsoever. Hence, possible transmission errors inevitably propagate to higher layers and have to be handled separately.
When IEEE 802.11 [51] was introduced, backward compatibility was apparently very important because IEEE decided to use the same LLC and only modify the MAC part for the new wireless standard. However, as wireless communication is inherently unreliable, error control needed to be added to the new wireless MAC sublayer [51, p. 24], in order to be able to guarantee similar functionality to the upper layer protocols. In the case of IEEE 802.11, it has been decided to add retransmission capabilities for unicast traffic into the MAC mechanism. Strictly speaking, however, this integration violates the SoC principle, because the responsibility of MAC is deciding when to access a shared physical transmission medium, but not necessarily when to retransmit a frame.
What's wrong with this design? Obviously, tight coupling prevents one from easily separating both mechanism. Let's assume we would like to replace the random-access-based distributed coordination function (DCF) of IEEE 802.11 with a reservation-based scheme. In IEEE 802.11, this can't be easily done without damaging its error control capabilities. This is why TDMA extensions for IEEE 802.11 based NICs do not provide link layer error

control [122, 123]. In order to still be able to cater for lost and erroneous transmissions, they have to install another layer for error control on top of the link layer. Furthermore, it's difficult to disable acknowledgments for unicast traffic, in case reliable transfer is not needed. Although with some NICs it is technically possible to turn off automatic acknowledgements, it is not part of IEEE 802.11. It is interesting to note that such use cases would have been possible, had IEEE chosen a more flexible and configurable link layer protocol for IEEE 802.11. For example, the high level data link control (HDLC) [36] protocol, which was already available at that time, would have been an alternative. Both examples serve well to demonstrate that MAC and error control are conceptually unrelated: they vary according to the system requirements and application area and should, therefore, be implemented separately from each other.

4.2.3 Link Layer Applications

The third element of the proposed architecture are link layer applications (LLAs). They can be understood as problem-specific protocols that use the GDTP for exchanging messages between instances. In that respect, they are similar to ordinary user applications. However, the main difference is that they are connected to the link layer controller to exchange control information by means of events. Thus, LLAs provide an elegant way to decouple management-related and application-specific parts from the pure data transfer functions. This makes them an ideal candidate to realize DSA specific functions, such as link establishment and spectrum mobility, one of the main functional requirements of this architecture. To illustrate this, consider link establishment (or rendezvous) as an example. Any link establishment mechanism requires some kind of handshaking protocol that coordinates the establishment procedure between two or more nodes. This handshaking protocol could be implemented as a LLA that starts broadcasting beacons when a radio is initially turned on. If the LLA receives a reply for one of its beacons from another node, it triggers a specific event to signal the link layer controller that a connection has been established.

Moreover, LLAs provide an elegant means to decouple traditional management functions within the layer. For example, Section 4.3 describes a protocol design that separates the functions of a TDMA-MAC into multiple components.

It is worth mentioning that LLAs are optional components within the architecture. There is no restrictions as to how many LLAs can exist in any specific link layer configuration.

4.2.4 Link Layer Controller

The fourth component of the FLL architecture is the link layer controller. The controller can be understood as the brain of the architecture, as it interconnects the individual components. The controller itself consists of two parts: one generic part and one application-specific part. The generic part assists the application-specific part in implementing the dynamic system behavior. Both will be explained in this paragraph.

4.2.4.1 Modeling Dynamic Behavior

The controller's primary function is mediating and coordinating the interactions among LLAs and the rest of the architecture. Hence, the controller avoids unnecessary coupling and dependencies among architecture components by exploiting the SoC principle. From a software engineering perspective, this separation may be achieved through the use of the *mediator pattern* [124].

In the remaining part of this section, we will describe how dynamic link layer behavior may be modeled using state machines, how the radio capabilities are managed, and how other components can query the link layer controller to retrieve relevant information, such as the number of accessible channels.

For modeling the dynamic behavior of a specific link layer configuration, the FLL architecture employs hierarchical state machines (HSMs) that are executed within the controller. A HSM, like a UML state machine [125], extends a traditional FSM by several concepts, making it more expressive. These concepts include:

- **Hierarchical states:** HSMs allow states that are nested inside other states. That means that an event defined for a superstate is implicitly also valid for all of its substates.
- **Orthogonal regions:** Orthogonal regions add parallelism to state machines by allowing concurrently active parts.
- **Entry and exit actions:** In a HSM, each state is allowed to define an optional entry action that is executed when entering a state as well as an optional exit action that is executed when leaving a state.
- **Guard conditions:** Guards are boolean expressions associated to transitions that are evaluated during runtime. However, the transition is only executed if the guard is evaluated to *true*.

As in traditional FSMs, the most important elements in HSMs are events. An event is a signal that can be triggered from a component to notify another component about the occurrence of this event. Events can cause state transitions within a HSM, but not necessarily do so. The link layer controller handles events from internal and external sources. Internal events are events triggered from other components of the link layer, for instance after a radio reconfiguration has been completed. Another form of an internal event are timer events that can be defined inside the HSM to trigger periodic actions.

External events, on the contrary, are events that may be triggered from a component of the underlying radio platform, such as a spectrum sensing component which has detected an active PU. The system examples below illustrate the usage of HSMs for modeling the interactions among protocols using events.

4.2.4.2 Generic Controller Facilities

The main purpose of the generic part of the link layer controller is to assist the application-specific part in fulfilling its task. This is achieved by providing generic facilities that may be required by more than one link layer configuration and, thus, can be shared among those. The generic controller facilities may include the following functions:

- **Generic information storage:** The link layer controller serves as a generic information storage of the FLL architecture. Other components can use this facility to store certain information, such as a list of neighbor nodes, a set of radio configuration parameters, or possible channel configurations. Other components can query the controller to retrieve this information, process them, e.g., to determine the right transmit power and modulation scheme, and write them back. In other words, the generic information storage is part of the FLL architecture that manages the (system-specific) PHY layer capabilities of a radio. The generic information storage is accessed through the event interface.

- **Handling of shared events:** Shared events are used by multiple link layer configurations and, thus, can be shared among those. For example, this may include events to query the generic information storage, to determine the current operating channel of a PHY interface, or the set of available channels.

- **Radio reconfiguration:** The link layer controller is also capable of reconfiguring any part of the radio during runtime. This allows it to react upon events triggered through other components and, as a result of them, modify the operation characteristics of the radio. However, the controller is only responsible for configuring a certain parameter. It is the responsibility of the LLAs to negotiate the correct parameter value among peers. A radio reconfiguration may be either a one-time event or permanent. For example, a one-time reconfiguration could be to set the transmit power for a single frame. For this purpose, the radio configuration may be attached to the frame that requires the reconfiguration as meta data. If a component along the processing chain of the frame detects a reconfiguration request destined to it, it will modify its characteristic itself, but only once and only for this frame. In contrast, if the reconfiguration is permanent, it is carried out through the controller and is not changed until the next reconfiguration. Changing the operating frequency, for example, would most probably be a permanent reconfiguration.

4.3 System Examples

Having introduced the core components of the FLL architecture, this section describes three system examples to support the understanding of the proposed concept and to illustrate the interaction among multiple protocol components. We begin by using the IEEE 802.11 standard in order to provide an example of how an existing protocol may be mapped on the primitives of the FLL architecture.

4.3.1 Implementing a Commodity Link Layer

Consider a IEEE 802.11 [51] mobile station (STA) that wants to set up a connection to an AP. The AP's link layer would comprise a "beacon LLA" that uses the GDTP to periodically broadcast beacons for advertising its capabilities. The counterpart of this would be a "scanning LLA" in the STA's link layer configuration that receives AP beacons. Knowing an available AP, a separate "association LLA" running on both APs and STAs would be needed to exchange authentication request

and response frames. Once a STA is associated to an AP, user applications can directly communicate via the GDTP. Understandably, the low-level DCF would be implemented inside a media-dependent MAC component. QoS, like in IEEE 802.11e through traffic categories/classes (TC), is inherently supported through the scheduler inside the GDTP and by maintaining independent queues per TC. Adaptation algorithms, e.g., for adjusting data rate or transmit power on a per-packet basis, could be implemented inside a link layer controller and another "link adaptation LLA". The controller and the LLA would use the information storage capabilities to maintain and reconfigure the link and physical layer as requested.

4.3.2 Separating Concerns of a MAC Protocol

The second example also describes the implementation of a commodity link layer protocol but has a stronger focus on the separation of concerns inside a single protocol. For a better understanding, consider a traditional TDMA-MAC protocol. Such a protocol is often composed of several parts: one that manages the slot allocation (i.e., adding and removing users), one that manages the time or time differences between users, one that is responsible for correcting transmission errors, and one that actually enforces and controls the medium access. Even though these functions are closely related to each other, they may be realized independently from each other and, thus, may be reused in different TDMA protocols.

Following the FLL principles, the time slot allocation and time synchronization part of a TDMA-MAC could be implemented as a LLAs as well. A possible protocol architecture sketch is depicted in Figure 4.5. In this example, the TDMA-MAC component on top of the PHY layer only assures that each outgoing frame of a particular node is transmitted at the right time. However, the mechanism that allocates time slots and compensates for clock drifts between nodes is implemented outside of the MAC component itself. This TDMA management protocol employs the GDTP in order to communicate with other instances of the same protocol on other nodes. Internally, however, the interaction between management protocol and MAC mechanism takes place through events and is mediated through the link layer controller.

4.3.3 Extending an Existing Link Layer

The third example describes the extension of an existing link layer configuration and the ease in doing so compared to the traditional approach. Imagine a basic data transfer protocol operating on a statically configured radio channel has been successfully implemented. If this system was to be extended by a link establishment mechanism, one would have at least two options to accomplish this goal. The first one would be to create an entirely new protocol by integrating the link establishment functions into the existing data transfer protocol. This would lead to a monolithic protocol design that is difficult to implement, maintain, and reuse.

The second option would be a component-based design, as proposed by this thesis. The core of this design is the GDTP that would be complemented by an additional component (i.e., a LLA) that implements the handshaking protocol. This component may define its own PDU types but uses the same interface as any normal user application would do for transmitting them to other instances of the same protocol. However, the dynamics of the system (i.e., its behavior) would be modeled as a HSM that is executed inside the link layer controller. Transitions between states

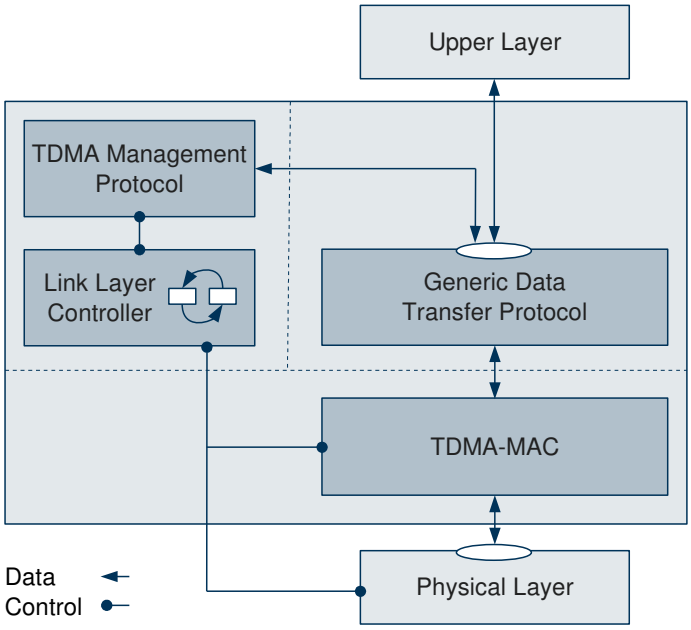


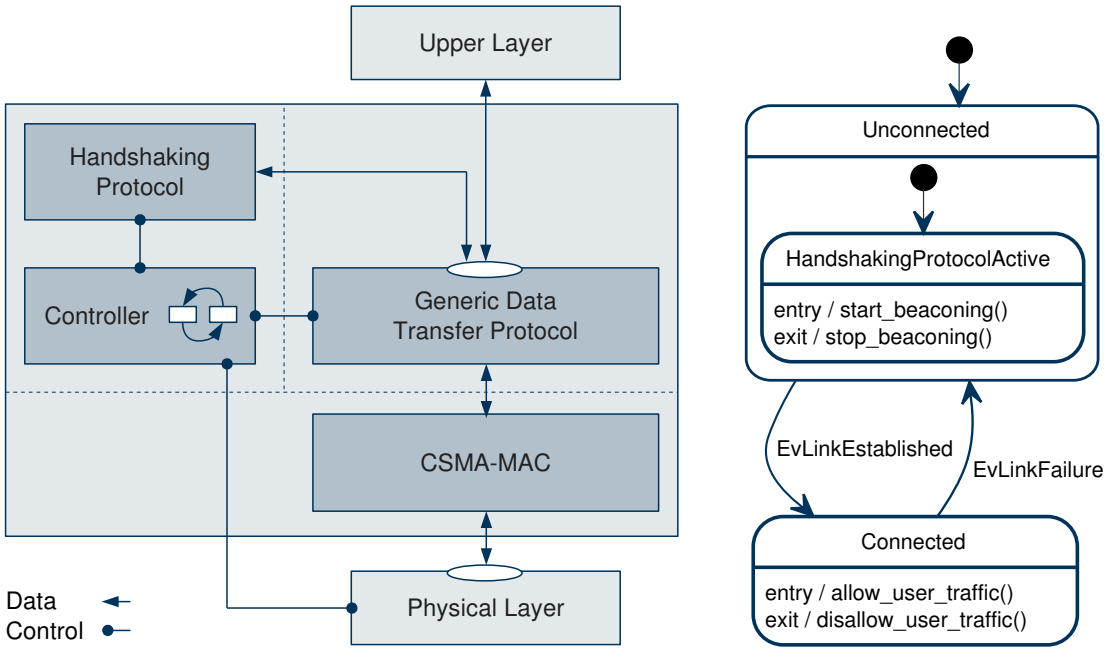
Figure 4.5: *Example of a TDMA-MAC whose slot management is implemented as a link layer application on top of the GDTP.*

of the HSM are caused by events that may be triggered through the LLA.

This example can be modeled with a HSM that consists of only two states (see Figure 4.6b). The system can be either in the *Connected* or *Unconnected* state. If a radio is initially turned on, it starts in the *Unconnected* state. The handshaking protocol is activated when entering the corresponding state (the `start_beaconing()` function is called). If a link could be established successfully, it triggers the appropriate event (i.e., *EvLinkEstablished*). This event causes the HSM to transit into the *Connected* state. If the HSM enters this state, the `allow_user_traffic()` function is called which signals the GDTP to resume serving flows that do not belong to the link layer itself, i.e., ordinary user applications. User traffic is again disallowed if a link is lost and a link failure is signaled, for example due to PU activity. This example shows how transitions between states of the HSM are triggered through an internal event issued by the handshaking protocol as well as an external event issued by the spectrum sensing component.

Figure 4.6 shows the components of this link layer configuration, its data and control connections, and the resulting HSM. It should be noted that starting and stopping the handshaking protocol could also be modeled with two nested states inside the *Unconnected* state.

In any case, the main advantage of following the proposed approach in this example is that both the GDTP and the handshaking protocol can be reused in other system configurations without reimplementing them from scratch.



(a) *High-level design with data and control connections.*

(b) *The HSM executed inside the controller.*

Figure 4.6: *Extension of an existing link layer by adding link establishment capabilities.*

4.4 Initial Analysis

As with any other new architecture, one question that immediately comes to mind is that of the associated trade-offs. How big is the communication overhead? How big is the component reuse gain? The purpose of this section is to provide an initial idea about the costs and gains of the FLL architecture, but without carrying out an in-depth quantitative analysis. Let's approach these issues from different perspectives: first, the communication overhead, second, the component reuse gain, and third, the overall complexity.

4.4.1 Communication Overhead

One may be concerned about the overhead of the componentization with respect to the amount of exchanged PDUs or extra header space. When introducing multiple modules at the link layer, i.e., LLAs, the only additional information required inside the PCI of the GDTP is the identification field that the multiplexer needs to deliver incoming PDUs (the SDU part of the GDTP-PDU to be precise) to the right recipient. Depending on the number of supported applications, the length of this field could range from 1 bit, i.e., bit set for upper layer application or bit not set for LLA, to several bytes. The actual processing costs of the multiplexer and for calling the LLA is implementation-specific. This could range from a simple function call to a context switch, if the LLA is implemented inside an own process. Please see Section 5.3.1.9 for a detailed overhead analysis of an actual GDTP implementation.

The FLL requires more PDU types than a conventional, monolithic architecture because the functions of a protocol are split across multiple components. One may be concerned about this additional overhead. However, as has been argued by Day [38], the actual communication overhead is relatively small for larger payloads. In fact, the overhead becomes almost negligible for frames whose payload size is multiple of the header size. For a more detailed description of this, including some historical background information, we refer the reader to Day [38, p. 76ff]. The overhead can be further reduced by allowing PDU concatenation. Interestingly, the overall overhead of empty or unused fields in protocols with fewer PDUs may be even higher, because they need to be transmitted in every single PDU.

One may also question the performance impact of the separation of MAC and error control. Let's assume a link layer configuration comprising a GDTP implementation with error control and a MAC and PHY similar to that of IEEE 802.11, i.e., with DCF-like channel access. From the viewpoint of this MAC, all data frames are handled like broadcast frames, i.e., without binary exponential backoff, as error control is done inside the GDTP on top of it. Hence, the performance of this configuration can be roughly compared to that of IEEE 802.11 in broadcast mode. Interestingly, it has been shown by Wang *et al.* [126], that the performance of unicast and broadcast traffic in IEEE 802.11 is comparable, even under high network contention. Even though those results cannot be directly applied to our configuration, because in our case, the PDU size would be larger due to the error control mechanism, they still provide an indication that the performance penalty can be tolerated.

4.4.2 Component Reuse

Flexibility and modularity are difficult to measure and compare. However, they are still best shown by looking at a practical example. Suppose we need to implement three different link layer configurations for three different network scenarios. The first configuration is quite trivial and only consists of a GDTP and a CSMA-MAC. This configuration doesn't require a controller. The second configuration is a bit more complex. It comprises a GDTP, a TDMA-MAC, and a spectrum mobility protocol. This configuration requires an individual controller to coordinate the spectrum mobility phase. The third configuration is the most complex one. It is based on the first configuration, i.e., it has a GDTP and a CSMA-MAC. In addition to that, it also has a link establishment and a spectrum mobility protocol. To coordinate the interactions, it too needs an own controller. All three link layer configurations are listed in Table 4.1.

To implement all three systems using the traditional protocol engineering approach, one needs to implement common functions multiple times, e.g., the GDTP and the CSMA-MAC. This would create a large amount of redundant code and would also complicate any modification to a component that is used in multiple configurations. In total, one would need to write nine protocols plus the additional code that implements the interaction logic: a total of eleven "software blocks".

By using a component-based architecture, one could reduce the amount of duplicated code. For example, the GDTP features only need to be implemented once, not three times. Hence, the total amount of components would be much less. To build the same system configurations, one would only need seven blocks in total: five protocols and two controllers.

In this example, the componentization of the FLL architecture reduces the amount of components and source code by 37%, compared to the traditional engineering approach.

Table 4.1: *Comparison of protocol architecture concepts in terms of required components.*

Component	Config 1	Config 2	Config 3	Traditional architecture	Component-based architecture
Controller		✓	✓	2	2
GDTP	✓	✓	✓	3	1
CSMA-MAC	✓		✓	2	1
TDMA-MAC		✓		1	1
Spectrum Mobility		✓	✓	2	1
Link Establishment			✓	1	1
			$\sum$	11	7

4.4.3 Complexity and Maintainability

Complexity reduction has been specified as a core architectural requirement in Section 2.4.4. However, at a first glance, adding more components to a system increases its complexity, which seems to be contradicting.

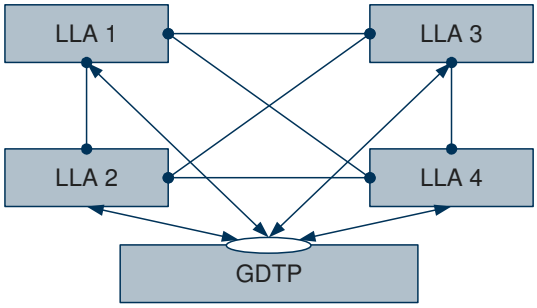
Admittedly, the decomposition of a complex system into smaller components and interconnections adds a certain amount of complexity beyond that of merely solving the issue of the complex system. Some system designers or project managers may even refuse such designs and characterize them as "too complex", something that Greer describes as "separation anxiety" [88].

In practice, however, we believe that a careful design and the additional complexity can be well justified for various reasons. First of all, it should be noted that componentization also removes complexity. This complexity reduction is mainly achieved by structuring the problem. In other words, it helps to transform a disordered complex system into an ordered complex system [88]. Instead of "hiding" the system's complexity inside a single component, the complexity of a component-based system usually lies in the interaction of the components with each other. The reduced individual complexity of components helps to better understand them, which is especially helpful for beginners in the field. It is also easier to distribute work among developers if a complex system is already modularized.

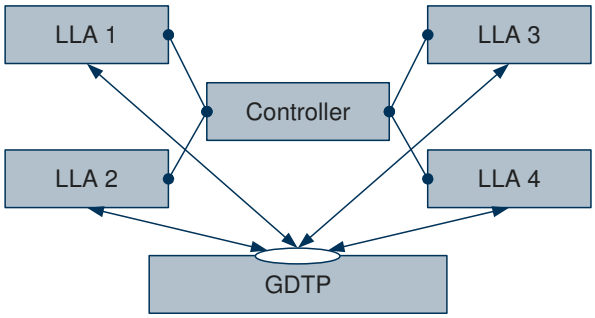
Second, componentization also reduces costs with respect to maintaining and extending a complex system. Considering the diversified application areas of FWCS, this advantage can't be overemphasized. Banker *et al.* [127] have analyzed the correlation between software complexity and maintenance costs. They found that software maintenance costs are highly affected by the level of software complexity. Specifically, it was found that maintenance costs increase with the complexity of a system's implementation, as measured by its average procedure size, average module size, and its branching complexity.

One may also ask whether the introduction of the link layer controller is worth the effort. We believe this is the case because of the increased reusability that it provides for the rest of the components. For example, consider a complex link layer configuration that consists of four protocol components, as shown in Figure 4.7. In this configuration, we assume that each component requires at least one service from each of the components (worst case scenario). Without using

a mediator object, i.e., the controller, the only way to achieve this requirement is by directly connecting the components with one another, as can be seen in Figure 4.7a. By using an additional object that mediates the communication between the protocol components, one can significantly reduce the total number of connections, as can be seen in Figure 4.7b. Even though this may be an extreme example, the link layer controller helps to realize flexible and reusable protocols, as will be shown in Chapter 5.



(a) *Without a controller object between link layer applications.*



(b) *With a controller object mediating between link layer applications.*

Figure 4.7: *Visualization of component connections with and without using a central controller (based on [128]).*

4.5 Summary

Due to the diverse set of application requirements and operating conditions, flexibility is of paramount importance for the development, deployment, and maintenance of FWCS. The SoC concept, whose main idea is to decompose a complex system into elements of non-repeating sets of cohesive functions, is a fundamental engineering principle that facilitates the development of flexible and reusable systems. Unfortunately, this principle is violated by a great number of communication systems, as has been pointed out in Chapter 3 of this thesis.

As a consequence, in this chapter, we have introduced the FLL architecture as a possible approach to address this problem. The fundamental concept behind the FLL architecture is twofold. First, the separation of data transfer and management-related functions. Second, the separation of media-independent and media-dependent functions. Within the architecture, this is achieved

through a component-based design that, at its core, consists of a generic, reconfigurable data transfer protocol. This protocol is not only used by higher layers for their communication needs, but also by link layer applications, which are responsible for management-related aspects of the layer. This allows to reuse and recombine existing components to entirely new, potentially unforeseen protocols, without the need to reimplement common functionality from scratch.

Initially, we raised the question whether the link layer of FWCS is fundamentally different to traditional systems. We found indicators suggesting that this is not the case, even though FWCS have to cater for more management tasks. In the specific case of DSA systems, this would allow us to realize link establishment and spectrum mobility as an application within the link layer architecture. But is this feasible? Can we really implement such an architecture?

The next chapter addresses these question by presenting a prototype of the proposed FLL architecture implemented on the basis of a practical SDR testbed. Furthermore, we will carry out a number of practical experiments to do both showcase the feasibility of the approach and to quantitatively evaluate the actual implementation.

5 Implementation and Evaluation

Having introduced the FLL architecture in the previous chapter, we will now transfer this abstract concept into an actual implementation in order to practically prove its feasibility and practicality.

In this chapter, the reader is taken on a journey that covers various generations of link layer protocols. This journey begins with a number of basic, general-purpose protocols. Those will then form the basis for successive generations of various highly adaptive link layer protocols for both fixed and dynamic spectrum access networks. In other words, we will see how a basic FLL configuration further evolves with each generation, as new requirements need to be met by the system.

For this purpose, we have implemented an experimental prototype of the FLL architecture on the basis of a reconfigurable SDR testbed. We employ this prototype to carry out a number of practical experiments in this chapter. To further complement the practical experiments when appropriate, we have also conducted various large-scale computer simulation. Before presenting and discussing the obtained results, we will first describe the evaluation methodology and experimental testbed.

Parts of the contents of this chapter have been presented at various conferences and workshops [5, 6, 7, 8].

5.1 Methodology

Determining the right means for evaluating network protocols is a difficult problem for various reasons. Analytical models, computer simulations, and practical experiments all have their own advantages and disadvantages.

In the research domain, the majority of protocols is evaluated either through analytical tools, e.g., Markov chain analysis, or through computer simulations. However, pure analytical studies can become quite complex and some problems may be quite difficult to investigate solely through mathematical expressions. In contrast, computer simulations often suffer from serious methodological problems or are unrepeatable, as has been reported by Kurkowski *et al.* [129]. But even when carefully designed, network simulations often do not have accurate PHY layer

Table 5.1: *Overview of evaluation methodologies.*

	Analytical model	Computer simulations	Practical experiments
GDTP	•	•	••
MAC	•	•	••
Link Establishment		••	•
Spectrum Mobility		••	•

Definitions:
- •• Main means for evaluation
- • Subsidiary means for evaluation

models, either because they are simply not implemented or too complex to achieve acceptable performance [101]. Furthermore, simulations often make assumptions that are extremely difficult to achieve in real-world systems, such as perfect time and phase synchronization among a set of distributed nodes.

As a consequence, we believe that practical experiments should complement any communication protocol evaluation study. Even if often small-scale due to the complexity of the experiment, proof-of-concept implementation is a powerful tool not only to "identify design problems, flesh out implementations details, and reveal limitations" [130], but also to, ultimately, prove it works.

Therefore, when appropriate, we have used analytical models to assess the performance of the protocols. In addition to that, we have used computer simulations, primarily to obtain long-term, large-scale figures. Furthermore, much attention has been devoted to developing prototypes in order to also carry out real-world experiments. Table 5.1 provides an overview of the different evaluation methodologies employed in this chapter. Before describing our experiments in detail, we first describe the evaluation platform.

5.2 Experimental Platform

This section describes the hardware and software platform that we used for our experimental evaluation. The entire software platform is based on freely available open-source tools. All software components that have been developed during the work on this thesis will also be made available under an open-source license. This allows researchers to easily reproduce the entire experiments, to extend the protocols, and to base new research on the findings of this dissertation.

5.2.1 Hardware Platform

The basic layout of the hardware platform is a classical host-based SDR configuration, as has been discussed in Section 2.5. As host computers, we used off-the-shelf *Lenovo ThinkPads* featuring an *Intel Core i5-2450M* clocked at 2.5 GHz and 4 GB of memory.

Each SDR node is equipped with (at least) one *Ettus USRP N210* [31] plus RF daughterboard. As daughterboards, we primarily used the *XCVR2450* dual band, half-duplex board. This board operates in the 2.4 GHz and 5.9 GHz ISM band. This setup is used throughout all practical experiments presented in this chapter.

The same setup was also used for a number of presentations and demonstrations at various conferences. For example, a demonstration showcasing a reconfigurable DSA network has been presented at the *10th International Symposium on Wireless Communication Systems (ISWCS)* in Ilmenau [8]. Table 5.2 provides an overview of the hardware configuration and basic system parameters. Specific MAC protocol parameters are explained in more detail below. For precise time and frequency domain measurements, we have used a *Rohde & Schwarz FSVR13* real-time spectrum analyzer. Figure 5.1 shows a photograph of our laboratory. The left hand side of the photograph shows two SDR nodes, the hosts in the back and the USRPs in the front. Both USRPs are time synchronized using a MIMO cable (see below). The spectrum analyzer and a vector signal generator are shown on the right hand side of the picture.

Figure 5.1: *Photograph of the laboratory and experimental testbed depicting two SDR nodes and the signal generation and measurement equipment.*

Time Synchronization Setup

For all experiments that required a TDMA-MAC, we needed to provide a common time reference to all nodes. This is required because the MAC itself doesn't cater for time differences and clock drifts between nodes. In order to still be able to precisely schedule frame transmissions, we used a common time source, a *Hameg HM8131-2* arbitrary function generator in the specific case, that distributed a 10 MHz reference signal as well as a 1 Hz pulse per second (PPS) signal. Both signals are distributed to two USRPs using a power splitter. The third USRP was synchronized to the second USRP using a multiple-input and multiple-output (MIMO) cable [131]. The MIMO cable is an expansion cable to connect a pair of USRPs together. This cable may be used to either connect two USRPs to a single host computer and/or to synchronize clocks between two devices.

Table 5.2: *Basic hardware configuration and system parameters.*

Parameter	Value
Host computer	Lenovo ThinkPad L520
RF hardware	USRP N210 + XCVR2450
UHD version	3.6.2
Sampling rate	2.5 Msamples/s
Carrier frequency	5.735 GHz
Channel bandwidth	2.5 MHz
Protocols	Ethernet, IP, UDP
Frame size	100 – 6000 B

Unfortunately, our setup is only able to provide the reference signals to three USRPs. Therefore, all experiments involving a TDMA-MAC are also limited to three nodes. Figure 5.1 only shows the MIMO cable but doesn't include the entire synchronization setup.

5.2.2 Software Platform

The operating system installed on the host computers was a standard Ubuntu 14.04 LTS with Linux kernel 3.2. In order to interface the USRP devices, we used the USRP hardware driver (UHD) version 3.6, which is a user-space library provided by the device manufacturer. The radio itself has been built using *Iris*, which is a user-space application implemented in C++. Compared to other SDR development frameworks, such as *GNU Radio* [108] or *Redhawk* [132], *Iris* is more suitable for link layer development because it allows components to directly communicate with one another through events. Furthermore, *Iris* is not limited to the stream-oriented processing model that is the central execution paradigm of most other frameworks. However, it is noteworthy to say that the message passing API that is available in recent GR versions provides a mechanism similar to *Iris'* events.

Even though *Iris* provides a number of signal processing blocks that can be used to built an entire radio, including single and multi-carrier modulators and demodulators, we have used a dedicated DSP library called *liquid-dsp* [133] to implement the PHY layer of the experimental platform. The main reason for doing so was the experimental and research-oriented nature of *Iris'* DSP blocks. For example, *Iris'* OFDM blocks have been developed with the main intention to investigate and advance PHY layer concepts, such as reducing out-of-band emission [134]. As such, throughput optimization or robustness was not a primary concern.

Even though *liquid-dsp* is a research project itself, its underlying DSP modules have proven to be more reliable and powerful for our experiments. In order to use *liquid-dsp* within an *Iris* radio configuration, we have developed a number of additional *Iris* components that interface the library. They are now part of the official *Iris* distribution.

In order to produce network traffic for benchmarking the entire SDR platform, we have relied on *nuttcp* [135], a free network performance measurement tool. We have used *nuttcp* to generate UDP traffic with a constant bit rate (CBR). The generated traffic is fed into *Iris* through a virtual

network driver. Figure 5.2 shows a sketch of the entire hard- and software platform.

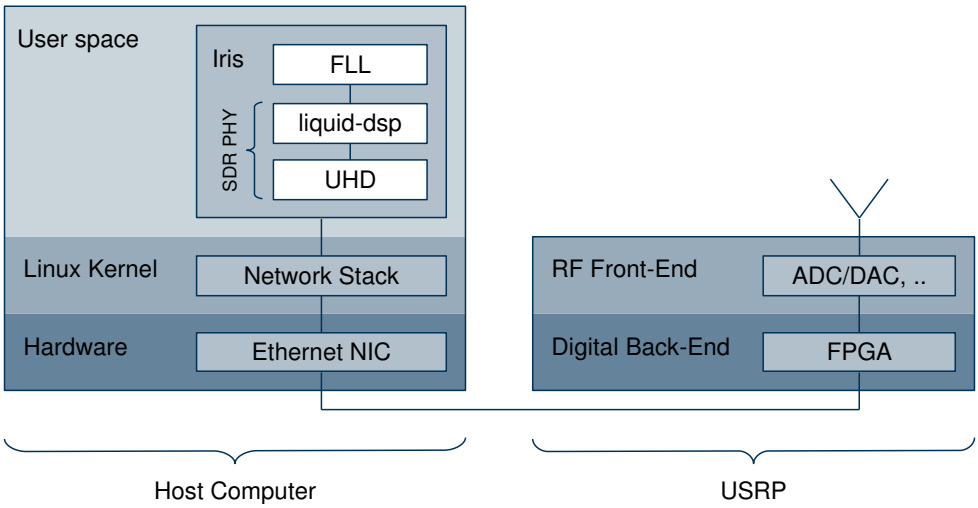


Figure 5.2: *Sketch of the experimental SDR platform.*

5.2.2.1 Physical Layer Configuration

As has been mentioned above, *liquid-dsp* [136, 133] has been chosen to implement almost the entire PHY layer of the experimental platform. In a nutshell, *liquid-dsp* is a modular, exceptionally well-documented and well-written open-source DSP library for SDRs developed in pure C. The library contains modules for mathematical operations, including windowing functions, matrix calculations, audio processing, various filter, error-correction codes, optimization routines, and an extremely versatile modem module. In our experiments, we specifically used the flexible OFDM frame generator and synchronizer provided by *liquid-dsp* to create an OFDM-based wireless communication system. The synchronizer allows to define a large number of OFDM parameters, such as the number and allocation[1] of subcarriers, their modulation scheme, the cyclic prefix, and the FEC code.

The spectral efficiency of a digital communication system, measured in bit/s/Hz, is defined as "the ratio of the bitrate to the bandwidth required to transmit that bitrate" [137]. In an OFDM-based communication system, each transmitted frame carries a constant amount of overhead. This additional information is needed for synchronization and demodulating the frame at the receiver. The spectral efficiency of a given system therefore depends on the size of the overhead relative to the transmitted user payload. The larger the payload, the lower the relative overhead and, thus, the higher the spectral efficiency.

Figure 5.3 illustrates the spectral efficiency of our PHY layer configuration as a function of the payload size. In our experiments, we have used *liquid-dsp* with its default OFDM parameters, i.e., we have not optimized the system for any specific scenario. From Figure 5.3, it can be seen that the spectral efficiency is relatively low for small frames. For example, with a payload size of 100 B, the system's spectral efficiency is only 0.4 bit/s/Hz. This is because it uses a relatively large

[1]The subcarrier allocation defines the number of carriers used for data, pilot and guard transmission.

number of pilot and null carriers, and a long cyclic prefix. The system has an asymptotic PHY efficiency of 0.72 bit/s/Hz. In other words, with a sample rate of 2.5 Msamples/s and a resulting channel bandwidth of 2.5 MHz, the maximal achievable throughput is 0.72 bit/s/Hz $\cdot$ 2.5 MHz = 1.8 Mbit/s. Throughout the chapter, all measurement results are normalized to 1.8 Mbit/s.

The detailed OFDM configuration parameters used throughout the entire experimental evaluation are listed in Table 5.3. For a more detailed explanation of the framing parameters and the features of *liquid-dsp* in general, we would like to refer the interested reader to the excellent tutorial section in [133].

Table 5.3: *Configuration parameters of the OFDM frame generator.*

Parameter	Value
Modulation scheme	QPSK
Subcarriers (total/data/pilot/null)	64 / 44 / 6 / 14
Tapering window length	4 samples
FEC (header)	-
FEC (data)	Hamming(12,8), code rate 2/3
Cyclic redundancy check	CRC32
S0 symbols	2
S1 symbols	1
Header symbols	7
Data symbols	variable
Asymptotic PHY efficiency	0.72 bit/s/Hz
Asymptotic PHY throughput	1.8 Mbit/s @ 2.5 MHz

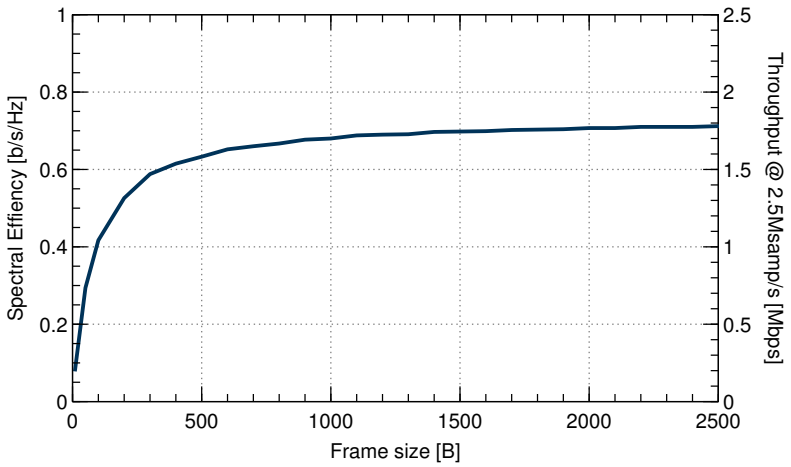


Figure 5.3: *Asymptotic spectral efficiency of the OFDM-based PHY layer used for the experiments. The detailed configuration parameters are listed in Table 5.3.*

5.3 General Purpose Link Layer Protocols

After having introduced the experimental platform, this section describes our efforts in transferring the idea of a generic data transfer protocol with decoupled medium access mechanism into reality. Specifically, we present the implementation details of *libgdtp*, our GDTP prototype library, and two fundamental MAC schemes for *Iris*. Furthermore, we discuss the details of various experiments.

5.3.1 libgdtp: A Generic Data Transfer Protocol Implementation

This section provides a detailed description of *libgdtp* [138], a prototype implementation of the generic data transfer protocol (GDTP). From this point onward, we will use the term *libgdtp* rather than GDTP to differentiate between our actual implementation and the conceptual description of the protocol in Chapter 4.

Rather than providing the entire set of protocol feature described in the previous chapter, we have concentrated our efforts on implementing only those concepts that are needed to carry out the practical experiments. In particular, relaying, concatenation/segmentation as well as security related protocol aspects are out of scope of this analysis. However, special care has been taken during the design in order to guarantee that further extensions can be done easily.

We will now describe each feature of the library and how it has been integrated into *Iris*. Furthermore, we provide a comparison of the overhead for two network scenarios in which the protocol may be used. We start with some general remarks on the design and implementation of the library.

5.3.1.1 General Remarks

The GDTP prototype, *libgdtp*, is designed and implemented as a shared user-space library. It is written in C++ and makes use of the standard template library (STL), e.g., for its basic data structures, and the Boost C++ libraries [139], e.g., for its threading and timer management. Logging inside *libgdtp* is implemented using *log4cxx*, a free logging framework for C++ [140]. The implementation follows the *GCC C++ coding conventions* [141] and is provided as free and open-source software under the GNU GPL license version 3 [142]. It is designed such that it can be used for both simulations and real-world practical systems.

Figure 5.4 shows a simplified class diagram of the library including all major components. For the sake of clearness, classes that are used inside the entire library, i.e., `class pdu`, `class exception`, and `class logging`, are not connected to each other in the graph.

5.3.1.2 Initial State Synchronization and Policy Selection

Libgdtp employs a timer-based mechanism to initialize the shared state between two communicating protocol instances. Upon reception of a data PDU that can't be assigned to an existing flow, a new flow with a default configuration is created automatically. By default, *libgdtp* runs in acknowledged mode (see below) and provides a reliable data transfer service between its clients. This reduces the overhead to a minimum before actual data can be exchanged. Therefore, a regular data exchange only requires two frames, one for the data to be sent and one for the

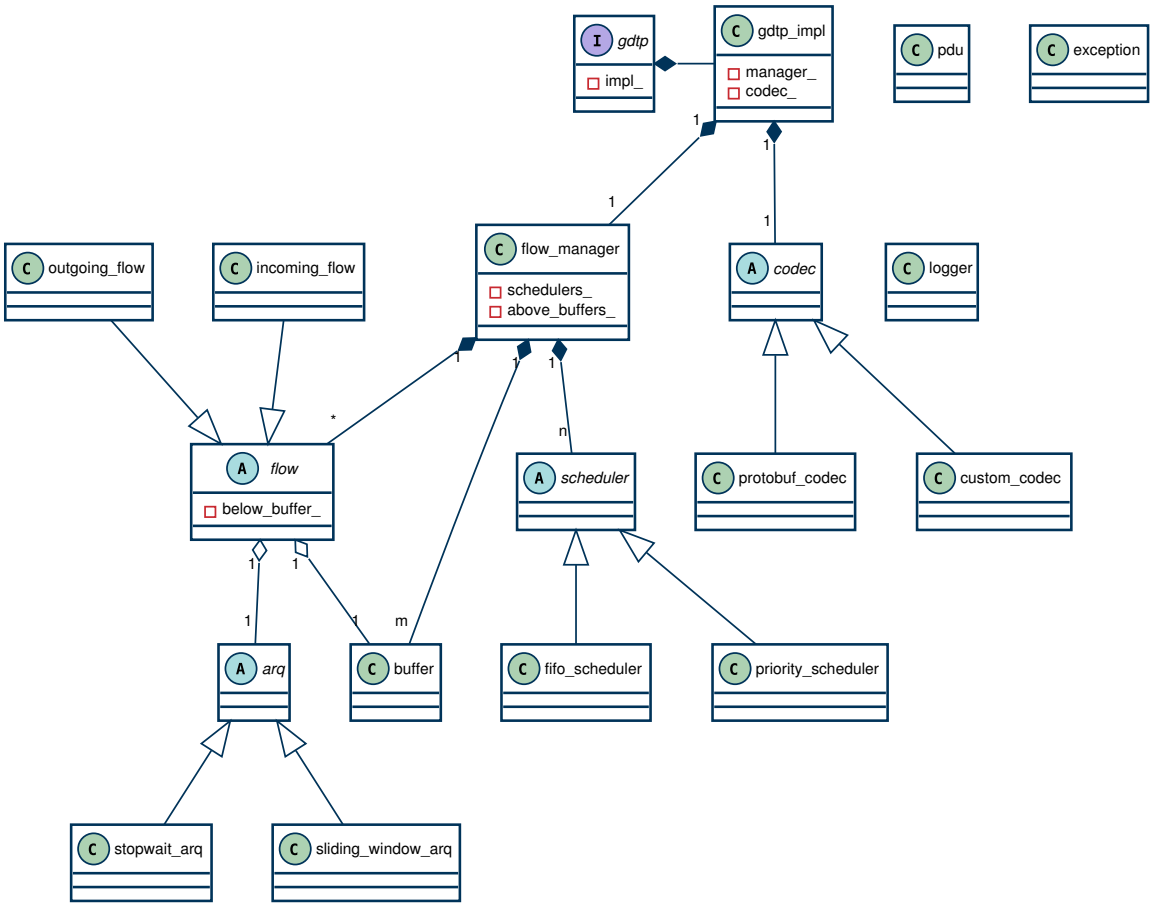


Figure 5.4: *Simplified class diagram of libgdtp including all major components of the library.*

acknowledgment. In fact, this strategy has been adopted from Watson's *Delta-t* transport protocol [143].

In Chapter 4, it has been mentioned that each modifiable parameter also requires a mechanism to change the value of that parameter during communication. Instead of implementing a separate mechanism for each parameter, the approach currently taken within the FLL architecture, it is conceivable to provide a single generic mechanism for this purpose, as has also be found by Kohler *et al.* [120]. The realization of such a generic mechanism, however, is left for future research. The interested reader is referred to Renker [144], who provides considerable background knowledge about the difficulties of changing communication parameters in the middle of a connection.

5.3.1.3 Interface

As has been stated above, *libgdtp* currently only implements the functions needed for experimental validation. As a consequence, only a subset of the intended services is exposed to the user of the library. Currently, the interface includes methods for initializing the library, setting default

parameters and basic flow requirements, and for passing frames up and down to the upper or the lower layer, respectively. Even though different flow requirements are used by a number of radio configurations, the current implementation requires parameters to be selected by both endpoints at the beginning of the communication. This is because feature negotiation for *libgdtp* during runtime is not implemented currently. Listing 5.1 shows a selected set of methods provided by the public interface of *libgdtp*.

```
// generic methods
void initialize();
void set_scheduler_type(const std::string type, const Port port);
void set_default_source_address(const Addr source);
void set_default_destination_address(const Addr destination);

// flow creation and manipulation
int allocate_flow(const FlowId id, FlowProperties props);
void modify_properties(const FlowId id, FlowProperties props);
FlowStats get_stats(const FlowId id);

// for upper and lower layer
void handle_data_from_above(std::shared_ptr<Data> data, const FlowId id);
void handle_data_from_below(const PortId id, Data& data);

// for lower layer transmitters
bool has_data_for_below(const PortId id);
void get_data_for_below(const FlowId id, Data& data);
void set_data_transmitted(const FlowId id);

// for upper layer receivers
bool has_data_for_above(const FlowId id);
std::shared_ptr<Data> get_data_for_above(const FlowId id);
```

Listing 5.1: *Basic methods provided by the public interface of libgdtp.*

5.3.1.4 Addressing

For identifying the source and the destination of a PDU, *libgdtp* employs 64 bit wide addresses. It supports three different types of addressing: in its standard configuration, user-modifiable default addresses are assigned to newly created outgoing flows. Therefore, it is mandatory to set a default local address during initialization of the library. If no destination address is specified explicitly, a broadcast address is used as the default destination address.

The second type of addressing is explicit addressing. During allocation of a new flow, the caller may explicitly specify source and destination addresses of the flow. If the provided source address has not been used previously, it is registered as a valid address for incoming data traffic. The registered addresses are then used for all outgoing PDUs of this particular flow.

The third type of addressing supported by *libgdtp* is implicit addressing. As the name suggests, addresses are not explicitly assigned to flows, but are implicitly provided by an upper layer protocol. Currently, *libgdtp* supports the IEEE 802.3 Ethernet standard [121] as an upper layer protocol. Why is this useful? When a new SDU is passed down from an upper layer protocol instance and the flow is configured to use implicit addressing, the SDU is processed as an Ethernet frame. Hence, source and destination address for all outgoing PDUs of this flow are taken from the corresponding fields of the Ethernet header. Implicit addressing in combination with a virtual network device, i.e., a TUN/TAP device under Linux, can be used to connect *libgdtp* to the networking stack of the operating system and allows to tunnel arbitrary network traffic through the SDR. In fact, implicit addressing has been used in a number of projects and practical demonstrations, for example for live video conferencing [145, 8] within a DSA network. Furthermore, implicit addressing is also used in all experiments in order allow *nuttcp* to transmit and receive its data transparently through the SDR. However, it should be noted that this type of addressing has the drawback that it requires Ethernet and IP headers and, therefore, also creates more overhead than explicit addressing.

5.3.1.5 Flow Identification and Multiplexing

In order to distinguish multiple flows between the same source and destination, a 64 bit wide flow identification field is used. This identifier is needed to deliver incoming PDUs to the right upper layer protocol or application.

Similarly, each flow is assigned a unique lower layer port that is used to deliver outgoing frames to the right lower layer protocol instance.

5.3.1.6 Error and Flow Control

The current implementation of *libgdtp* supports two operation modes that provide a default selection of policies for error and flow control:

- **Acknowledged mode (AM):** In AM, lost and duplicate detection is enabled. A simple stop-and-wait ARQ is used by default with a credit scheme of size one as flow control mechanism. Frames are delivered in order.
- **Unacknowledged mode (UM):** In UM, only duplicate detection is enabled. Error as well as flow control are disabled. However, frames are also delivered in order.

5.3.1.7 Scheduling

Two frame scheduling algorithms are currently implemented within *libgdtp*. The default scheduler is FIFO-based. This scheduler queues flows that are marked ready and processes them in the order in which they arrived.

The second scheduler supported by *libgdtp* is a preemptive, priority-based scheduler that arranges flows inside the ready queue in order of their priority. The priority of a flow, a 32 bit wide integer value, can be defined during its creation. The lower the value, the higher the priority of the flow. As scheduling is done on a per-frame basis, lower priority flows can get preempted by the scheduler when higher priority flows become active. In fact, this feature is exploited within

the FLL architecture in order to give traffic originating from LLAs preference over traffic generated from ordinary user applications.

5.3.1.8 Protocol Data Unit Format and Serialization

PDUs inside *libgdtp* are modeled as an own class called `class pdu`. This class contains the PCI as member variables and uses a shared pointer to store its actual SDU, if any, in a memory-efficient manner. Handling of PDUs inside the core library is done entirely based on objects of this class.

However, before PDU objects are sent from one protocol instance to another, they need to be serialized (or encoded) into an array of bytes. On the receiving node, this array again needs to be deserialized (or decoded) into a valid PDU representation, i.e., an object of `class pdu`.

In *libgdtp*, we have decoupled the internal representation of a PDU and the actual format that is used for over the air transmissions in order to allow modifying the latter without touching the former. The en- and decoder are capsuled within an object of an abstract class called `class codec`.

The default serialization format used in *libgdtp* is *Google Protocol Buffers* [146] (or short, *Protobufs*). A *Protobuf* en- and decoder is implemented in `class protobuf_codec`. Listing 5.2 shows the description of the *Protobuf* message that is used to encode a `libgdtp` frame, i.e., an object of `class pdu`. *Protobuf* is a serialization format that uses variable-length encoding to optimize the space needed to transfer an encoded message.

Traditionally, headers of communication protocols use fixed length fields. This allows efficient parsing of incoming frames but requires to define the width of each field at design time. For example, the maximum number of addressable nodes is defined by the length of the address field.

However, the maximum number of nodes in a network deployment may not be known at design time. This can lead to an address field that is either too short or too long. If the address field is too short, the maximum number of addressable nodes may be too small. On the other hand, if the address field is too long, valuable space in the protocol header is wasted with every transmission. By using an encoding scheme with variable-length protocol fields, such as *Protobuf*, this issue can be mitigated and valuable header space can be saved. After the next section, we will study the efficiency of *libgdtp* in greater detail and also compare it to conventional protocols using fixed-length encoding.

Figure 5.5 illustrates the processing of an upper layer SDU within *libgdtp*. In this example, an upper layer protocol transmits 1500 B of payload. *libgdtp* adds its own PCI to the SDU received from the upper layer protocol. The entire *libgdtp* PDU is passed down to a lower layer protocol.

5.3.1.9 Protocol Overhead Analysis and Efficiency Simulation

This section evaluates the overhead of *libgdtp*. Specifically, we compare the overhead when communicating in AM where each frame is acknowledged individually. We compare two different network scenarios:

- **Small network scenario:** This network scenario comprises relatively few nodes (i.e., ≤ 128) that only require low-bandwidth communication.

```
// libgdtp Protobuf message description
message GdtpPdu {
    enum PduType {
    DATA = 0;
    ACK = 1;
  }
  required uint32 src_id = 1;
  required uint32 dest_id = 2;
  required uint64 source = 3;
  required uint64 destination = 4;
  optional uint64 seqno = 5;
  required PduType type = 6;
  repeated bytes payload = 7;
}

message GdtpFrame {
  repeated GdtpPdu pdu = 1;
}
```

Listing 5.2: *Description of a Protobuf message to encode a libgdtp PDU.*

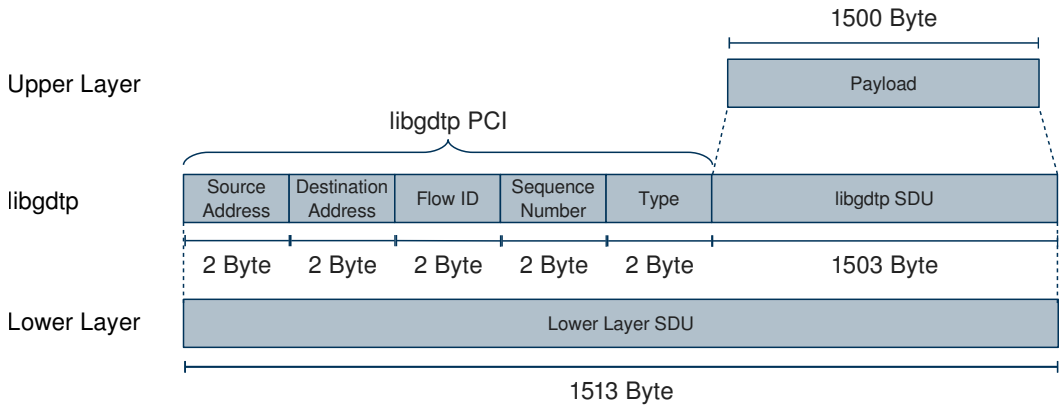


Figure 5.5: *Processing of an upper layer protocol payload within libgdtp.*

- **Large network scenario:** This network scenario comprises a potentially large number of nodes (i.e., up to 2^{64}) that also require high-bandwidth communication.

Apart from two network scenarios, we also assess varying channel conditions by simulating a specific target FER. Our analysis shows that the same protocol implementation can be used in both scenarios but achieves a much better efficiency compared to state-of-the art protocols with fixed-length protocol fields.

The reason for differentiating between small and large networks is the difference in space in the protocol header required to store node addresses as well as port identifiers and sequence numbers. With fixed-length protocol fields, which are used in most communication protocols

(e.g., TCP or UDP), valuable space inside the headers is wasted if only small numbers need to be encoded. This is the case if a network consists of only a few nodes. The same is true for sequence numbers: in small networks with a low BDP, encoding the sequence number of each frame inside a fixed-length 64 bit integer may be inefficient because there will never be that many *active* frames at the same time. In contrast, it has been found that data corruption through duplication of sequence numbers can happen in today's high-speed networks due to a too small sequence number window. At a transfer rate of 1 Gbit/s (or 125 Mbit/s), the 32 bit sequence number used in TCP may be wrapped in 17 s [147], which is below the maximum segment lifetime (MSL) of 120 s that is needed to guarantee the reliability of TCP.

Performance Metric

We use protocol efficiency as a metric for comparison. The efficiency η (see Equation 5.1) accounts for the overhead O (in bytes) introduced by the protocol in order to transfer a certain number of payload bytes N from node A to B, i.e., it includes the overhead O_A that the transmitter may add to the payload as well as the overhead O_B that the receiver may add to signal the reception of the payload. Thus, η may be calculated as follows:

$$\eta_{AB} = \frac{N}{N + O_A + O_B}. \tag{5.1}$$

Overhead Analysis

Table 5.4 shows a detailed comparison of the overhead induced by the transfer protocol for sending 1500 B of user data in AM between two network nodes. In this particular example, we assume an error free channel, i.e., no retransmission due to lost data or acknowledgment frames.

As has been mentioned above, *libgdtp* employs 64 bit source and destination identifiers internally. In order to encode those using Protocol Buffers, $2-9$ B are needed depending on the actual number. To encode integer numbers between 0 and 127, i.e., the addresses of our small network scenario, only 2 B are needed. Note that even though the actual number can be encoded in only one byte, *Protobufs* need an additional byte in order to encode the field number (defined inside the message definition) and the wire type. The wire type is needed for detecting and decoding the correct data type. For example, a variable-length integer or a length-delimited field, such as a string or an array of bytes, is encoded differently. This is also the reason why 1500 B user payload actually require 1503 B.

In this analysis, we assume the minimum length of 2 B for all protocol fields in the small network scenario and a fixed maximum length of 9 B for all addresses and the sequence number field in the large network scenario, respectively. As can be seen from the results in Table 5.4, the relative protocol overhead for sending 1500 B is ~1.7% in the small network scenario and ~4.9% in the large network scenario. In other words, for sending 1500 B over an error free channel, *libgdtp* is over 3% more efficient than similar protocols that use fixed-length encoding. We now expand this analysis through simulations. Specifically, we study varying payload lengths and also consider erroneous channels.

Table 5.4: *Comparison of the best-/ worst-case protocol overhead for sending* 1500 B *user data in AM in small and large network scenarios (Protocol Buffers serialization). All figures are in bytes, if not marked otherwise.*

Parameter	Small Network		Large Network	
	Data	Ack	Data	Ack
User payload	1500	-	1500	-
Source Address (2-9)	2	2	9	9
Destination Addr. (2-9)	2	2	9	9
Flow Id (2-9)	2	2	9	9
Sequence Number (2-9)	2	2	9	9
Type (2)	2	2	2	2
Encoded payload (2-*n*)	1503	2	1503	2
Sum	1513	12	1537	36
Overhead	13	12	41	40
Overhead [%]	1.64		5.12	
Efficiency [%]	98.36		94.88	

Efficiency Simulation

We have used the `benchmark` example that is part of *libgdtp* to run simulations with varying application payload lengths between $10 - 1.500$ B and a varying FER between $0 - 20$ %. In each simulation run, we transmitted 10.000 frames with a random payload pattern from an emulated user application on a transmitter protocol entity to an emulated user application on a receiver protocol entity. Note that we emulated varying channel conditions on a per-PDU basis, instead of on a per-bit basis. Therefore, the same FER has been used in both directions, i.e., for data and acknowledgment PDUs, even though in practice shorter PDUs usually have a lower FER than larger PDUs.

Figure 5.6 plots the efficiency under varying channel conditions as a function of the PDU length for both scenarios. Specifically, by comparing equally colored curves in both plots, we can observe how smaller frames benefit from the variable-length encoding technique compared to fixed-length encoding. For example, for a user payload of 60 B, the efficiency is improved by 23% in an ideal channel. In a channel with 20% FER, *libgdtp* is still 15% more efficient.

This example demonstrates the advantage of using variable-length over fixed-length encoding. In our case, the protocol overhead could be reduced significantly in some situations, for example when only few node addresses are sufficient. Even though the same could have been achieved by allocating fewer bits in the corresponding header fields, this would have destroyed the flexibility needed to use the same protocol for networks that consist of a much larger number of nodes.

In addition to that, due to the use of a timer-based state synchronization mechanism with default parameters, explicit synchronization through the exchange of request and response PDUs, such as a three-way-handshake in TCP, is not necessary. Therefore, assuming an error-free channel, a complete data exchange only requires two packets, one for the data to be sent and one for

the acknowledgment.

5.3.1.10 A libgdtp Component for Iris

As has been stated above, *libgdtp* was designed such that it may be used for simulations, as has been demonstrated in the previous section, as well as for real-world systems. In the context of this thesis, by "real-world system", we mean a SDR framework that implements the lower and upper layers of the system. As *Iris* is used for this purpose, we have designed and implemented a component for it, called *GdtpComponent*, that capsules *libgdtp* and provides its features to other *Iris* blocks.

Figure 5.7 shows the high-level design of a link layer configuration that only consists of the GDTP. As this link layer does not have a MAC component, the GDTP is directly connected to the PHY layer of the radio. For example, such a configuration could be used for point-to-point links that don't require a MAC mechanism.

5.3.2 SoftCsma: A Software Implementation of a CSMA-MAC for Iris

In the previous section, we have presented *GdtpComponent* as a wrapper for using *libgdtp* from within *Iris*. Even though this component can already be used in a practical system, its usefulness as a stand-alone protocol is limited in many realistic environments. This limitation is primarily because collisions are likely to occur on a shared medium if multiple STAs have data to send. To overcome this issue, there needs to be a mechanism that manages the access of multiple STAs on a shared medium: a MAC mechanism.

This section presents a software implementation of a MAC mechanism for *Iris* that has been developed with the goal to provide a simple, yet useful random-access control for multiuser SDR networks without needing external components or hardware modifications[2].

Having the challenges for implementing link layer protocols in mind, we will explore and extend the current understanding for executing delay sensitive protocols on conventional host-based SDR platforms. Specifically, we will show that CSMA-MACs exhibit a similar behavior across different platforms. In other words, it is demonstrated that a CSMA-MAC implemented on a host-based SDR behaves similar to commodity IEEE 802.11 hardware.

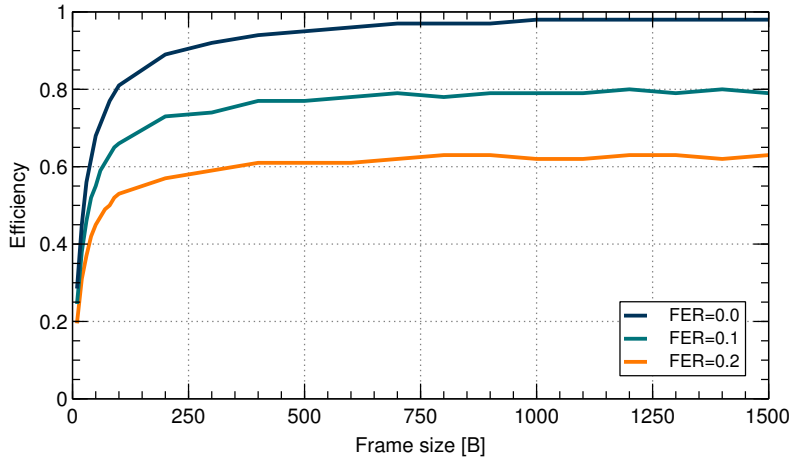
We begin with a description of the system and component architecture. In the following, we explain the implementation strategy and describe the protocol operation. We then evaluate the performance of the practical implementation that has been realized in our experimental testbed.

5.3.2.1 System and Component Architecture

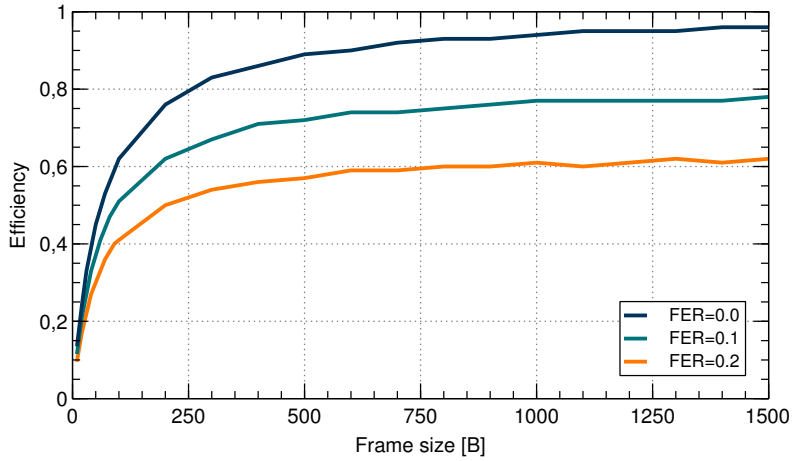
Following the SoC philosophy of this thesis, the functions of the CSMA-based link layer for *Iris* have been distributed across different components. Figure 5.8 shows the high-level design of the protocol on the basis of the FLL architecture comprising two main components:

- **Generic Data Transfer Protocol:** The GDTP component provides the basic link layer functions, such as error and flow control.

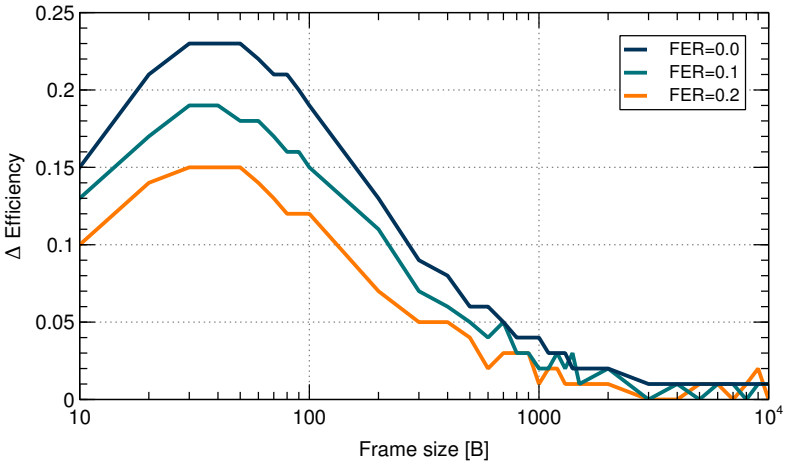
[2]For example, hardware modification could include changes to the FPGA logic of a SDR platform.



(a) *Small network scenario.*



(b) *Large network scenario.*



(c) *Difference between the small and large network scenario.*

Figure 5.6: *Simulated efficiency of libgdtp under varying channel conditions for small and large network scenarios.*

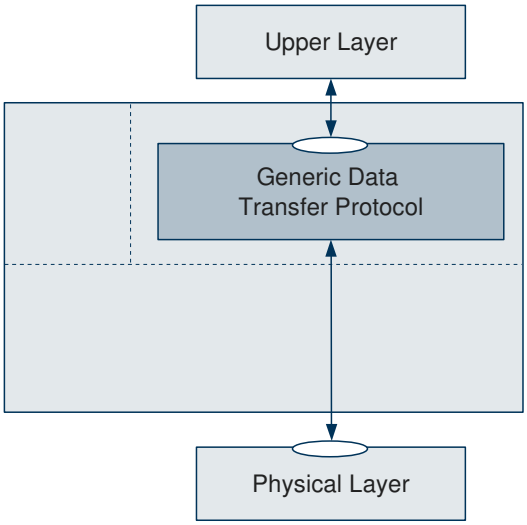


Figure 5.7: *High-level design of a link layer configuration for point-to-point connections on the basis of the FLL architecture.*

- **CSMA-MAC:** The CSMA-MAC component implements the "listen before talk" medium access strategy.

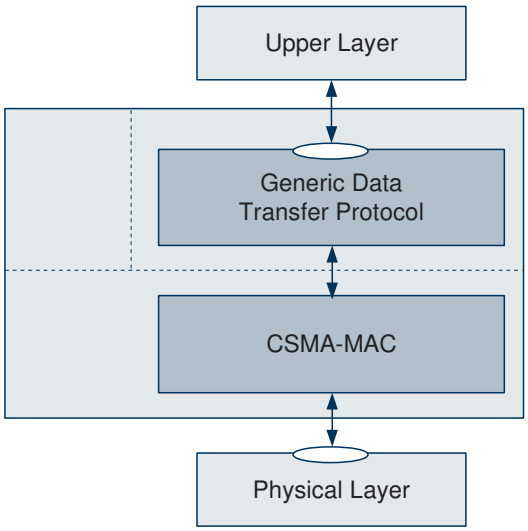


Figure 5.8: *High-level design of a CSMA radio on the basis of the FLL architecture.*

Note that this basic link layer configuration neither requires a LLA nor a link layer controller. However, in order to provide a fully working link layer with functions similar to those of IEEE 802.11, few more components are required. In fact, the CSMA-MAC component depicted in the high-level protocol design is a compound component itself. In other words, it consists of a number of other components that complement the core CSMA component and together provide the functionality of a CSMA-MAC. In particular, those components are responsible for realizing

the CCA mechanism. The implementation of this mechanism consists of two additional blocks:

- **Energy Detector:** This component implements a particular energy detection (ED) algorithm or capsules an external ED mechanism, such as an FPGA or any other hardware device.
- **CCA Coordinator:** This component coordinates the interaction between the core CSMA component and the ED block.

As a consequence, the actual CSMA link layer implementation consists of four components altogether. This separation has one big advantage over monolithic implementations: it allows to change a single component without modifying the remaining part of the architecture. In fact, this allows to replace the actual implementation of the ED component across different CSMA-MAC configurations. For example, we have demonstrated the possibility of using an external, high-performance sensing engine as ED within our architecture [13, 6]. Furthermore, we have also shown how to implement the entire MAC within the FPGA of the radio front end [6, 11]. In this thesis, however, we will only focus on the software version of the ED component[3].

Figure 5.9 illustrates the actual *Iris* flow graph of the software implementation. It comprises the *SoftCsma* component, an energy detector implemented in software, and the CCA coordinator[4]. In addition to that, it also includes the receiver and transmitter blocks as well as the modulator and demodulator. All components as well as the CCA mechanism will be described in the following paragraphs.

5.3.2.2 Principle of Operation

The heart of the CSMA-based link layer is the *SoftCsmaComponent* for *Iris*. This component mimics the behavior of the IEEE 802.11 DCF protocol including the binary exponential backoff. The component treats data and acknowledgment frames differently, based on the meta data that is passed along with the frame and activates/deactivates the CCA mechanism accordingly. Whenever a new data frame is passed down to the *SoftCsmaComponent*, the component enters the CCA state and issues a CCA request (see tag (1) in Figure 5.9) to the *MacPhyCoordinator*. This event is then passed over to the ED block, the *EnergyDetectorComponent* in our case (see tag (2)).

While CCA takes place, the component blocks and waits to continue operation according to the sensing outcome. After the ED component has completed the sensing process, it issues a CCA result event (see tag (3)) to the coordinator which includes the outcome of the sensing process (i.e., the computed energy level). Based on this value, the *MacPhyCoordinator* determines the channel status by comparing it with a predefined threshold. The channel state is then reported back to the CSMA-MAC as a Boolean value (see tag (4) in Figure 5.9). With this information, the CSMA-MAC decides whether to transmit the frame. As has already been noted above, the main advantage of the modular design is that it allows to implement the MAC logic independently from the actual CCA mechanism. Understandably, however, this architecture and its implementation in *Iris* also have an impact on CSMA protocol parameters, such as slot time, SIFS, and DIFS. We will discuss those later in this section during the experimental evaluation.

[3]Note that preamble detection [51], another method for CCA, is not discussed in this thesis.
[4]The actual *Iris* component is called *MacPhyCoordinator*.

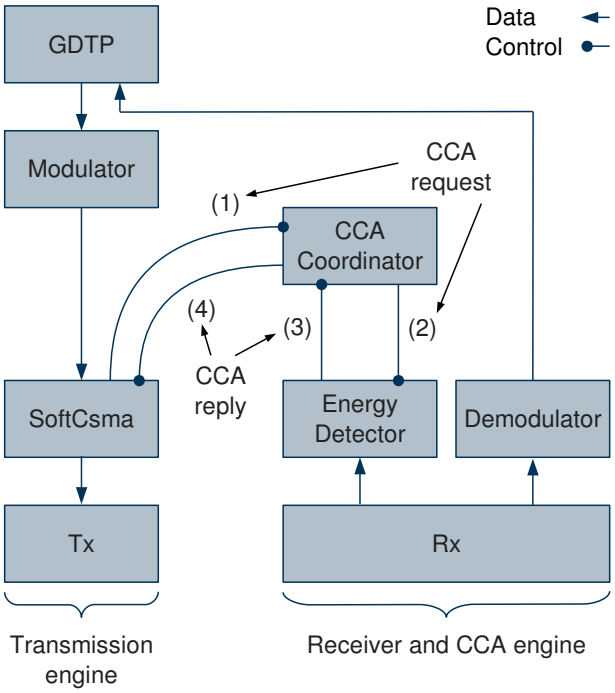


Figure 5.9: *Iris radio flow graph (simplified) of the SoftCsma implementation illustrating the CCA mechanism.*

5.3.2.3 Energy Detector

The fundamental principle behind CSMA-MACs is their CCA mechanism, which is used to avoid collisions by transmitting only when the channel is found to be idle. As a consequence, the radio architecture requires a component that is able to determine the state of the channel. For this purpose, we have developed the *EnergyDetectorComponent* that implements ED in software. ED performs a binary hypothesis test on whether a signal, i.e., another user, is present in a particular channel or not [23, p. 86]. The channel is reported to be idle under the null hypothesis and busy under the alternate:

$$\mathcal{H}_0 \text{ (idle)} \quad \text{vs.} \quad \mathcal{H}_1 \text{ (busy)}. \tag{5.2}$$

In the idle case, the received signal contains only the ambient noise in the radio frequency environment, while in the busy case, the received signal consists of the ambient noise plus the signal of the other user. Therefore, $\mathcal{H}_0$ and $\mathcal{H}_1$ can be written as:

$$\begin{aligned} \mathcal{H}_0 &: x(t) = n(t) \\ \mathcal{H}_1 &: x(t) = s(t) + n(t), \end{aligned} \tag{5.3}$$

for $t = 1, \ldots, m$, where m is the number of received samples, $n(t)$ the ambient noise, and $s(t)$ the user's signal. The decision rule is then given by:

$$y(t) \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \gamma(k), \tag{5.4}$$

where $\gamma(k)$ denotes the detection threshold [148]. The decision statistic for the ED itself is expressed as the average squared magnitude of the received signal [149]:

$$y(t) = \sum_{i=1}^{n} |x(i)|^2 . \tag{5.5}$$

The ED approach has been chosen for CCA due to its low complexity and applicability, even if no or only little prior knowledge about the transmitted signal is available. In our implementation, the *EnergyDetectorComponent* receives the incoming stream of I/Q samples and computes $y(t)$ for a specific time window and bandwidth as given above. The computed value is then provided to the *MacPhyCoordinator* for comparison with the specified threshold.

In practice, defining a single optimal threshold that achieves the desired performance characteristics is a difficult problem. Due to a high SNR in our laboratory, we were able to simplify this process. The *EnergyDetectorComponent* allows to monitor the received signal power. With this information, we set the threshold that gave us satisfactory detection performance between the power level of the noise floor and a frame transmission, without requiring a reference device for precisely calibrating the power levels. Figure 5.10 illustrates the power levels when transmitting a data and acknowledgment frame, as observed from the ED component.

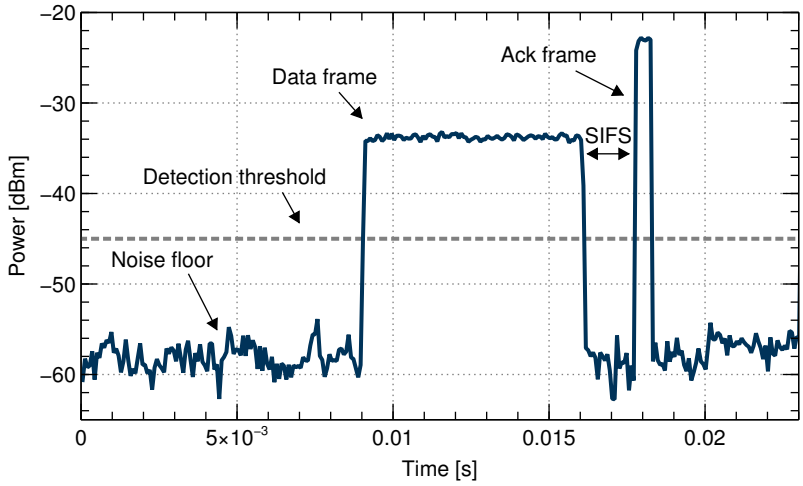


Figure 5.10: *Power levels when transmitting a pair of frames for setting the detection threshold of the ED component.*

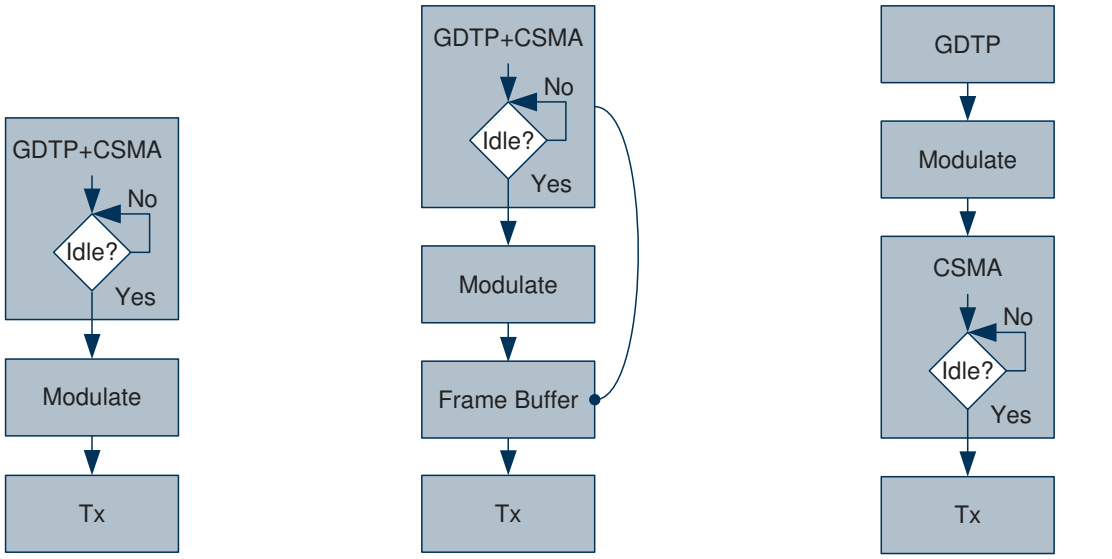
5.3.2.4 Implications of Component Separation on Implementation Complexity

This section argues that the separation of GDTP and MAC has a direct consequence on the implementation complexity of the link layer. Overall, we believe the separation reduces the implementation complexity significantly. Consider following two examples that support this argument.

Example 1: Reduction of Processing Delays in Host-based SDRs

The transmission of data frames using a CSMA-MAC is a complex process that requires a number of steps to be carried out sequentially. At first, a frame needs to be pulled off the transmission queue. It then needs to be framed and modulated. Before the frame can be sent, the radio must first observe an idle medium for a certain amount of time.

A straightforward software realization usually leads to a solution that implements all steps within a single component [116, 35, 9]. Using a conventional host-based SDR, however, strictly following this sequence may cause severe performance drawbacks due to the low signal processing capabilities of standard PC hardware. This is primarily because the frame needs to be modulated *before* it is actually sent. This takes a relatively large amount of time and increases the latency between the time when MAC observes a transmission opportunity and the actual time at which it accesses the medium. Figure 5.11a illustrates a simplified transmission flow graph (without backoff functionality) of an architecture with combined data transfer protocol and MAC mechanism. Note the modulator that sits between the protocol and the transmission block, which significantly increases the overall channel access delay of this approach.



(a) *Initial protocol architecture.*

(b) *Promodulation architecture with combined GDTP and MAC, frame buffer, and signaling mechanism (reproduced from [6]).*

(c) *Enhanced protocol architecture with separated GDTP and MAC.*

Figure 5.11: *Comparison of different SoftCsma architectures with and without separated GDTP and MAC. Each flow graph shows the processing sequence for transmitting a frame. Note that the receiver chain is not displayed.*

To overcome this problem, we proposed to modify the transmit flow graph of our initial architecture such that the frame is modulated *before* CCA is executed. Specifically, we proposed to introduce an additional buffer between modulator and transmitter [6]. This optimization strategy, called *premodulation*, effectively reduced the timespan between the time instant at which the shared medium is detected to be free and the actual access time. Figure 5.11b depicts the

premodulation architecture.

However, the proposed mechanism has a couple of shortcomings. First, the traditional execution model between MAC and PHY layer needs to be split and the implementation of the MAC component is considerably more complex. Furthermore, this solution requires an additional component, the frame buffer, and a complex signaling mechanism between this buffer and the MAC component. The purpose of the signaling mechanism is to trigger the transmission of a frame stored inside the frame buffer from within the MAC. We believe the main reason for this complex solution was a misunderstanding as to which functions really need to reside inside the data transfer protocol, and which should be placed outside of it.

As the architecture evolved and the split between data transfer and MAC was performed, it was interesting to observe performance benefits caused by an architectural decision. Using the SoC approach, we were able to achieve the same latency reductions with a much simpler software architecture. Figure 5.11c shows the enhanced architecture with decoupled components, as compared to our initial solution with tightly integrated mechanisms.

Example 2: Standard-compliant Implementation of IEEE 802.11

The first example illustrates well how the separation of GDTP and MAC may reduce the implementation complexity of the link layer in a SDR. However, from a protocol architecture point of view, it is interesting to understand the reason for first choosing the more complex solution. Admittedly, this question is difficult to answer but we believe the famous IEEE 802.11 standard influenced our thinking. Because MAC and error control are intertwined in IEEE 802.11, one is tempted to follow the same strategy. As we have seen above, this may result in a complex component architecture when implemented in software.

But what are the implications of this, if the protocol was to be implemented in a standard-compliant way using SDRs? It is widely accepted that implementing standard-compliant IEEE 802.11 unicast transmissions is challenging due to the temporal constraints of the protocol [34, 35]. Even though it has been shown to be technically possible [78, 75], it still requires quite some effort and a huge amount of computational resources. The main reason for that is again the combination of MAC and error control in IEEE 802.11.

When receiving a valid unicast data frame, a standard compliant IEEE 802.11 radio is required to respond an explicit acknowledgment within a certain amount of time, the SIFS period, in order to signal the successful reception of the data frame.

Figure 5.10 illustrates a unicast transmission between two STAs using our software implementation of the IEEE 802.11 protocol. But of course, this protocol doesn't satisfy the IEEE 802.11 timing constraints. In order to do so, one would need to implement a number of mechanisms in an efficient and deterministic manner. Those mechanisms include detecting an incoming frame, synchronizing on it, demodulating it, checking its correctness, framing the acknowledgment, modulating it, and transmitting it. In other words, almost the entire transmitter functionality needs to be implemented. Understandably, this requires a great number of computational resources and, therefore, is difficult to realize using low-cost SDR hardware with limited FPGA capacity, such as the USRP.

In contrary, it is perfectly viable to only implement the MAC mechanism without any GDTP functions on this class of SDR devices, as has been successfully demonstrated by a number of researchers [35, 11, 76]. A question that immediately comes to mind is whether there is a clear

performance benefit that justifies the implementation complexity of IEEE 802.11. For answering this question, one has to compare IEEE 802.11 with a protocol that provides similar features for unicast transmissions, but doesn't require explicit acknowledgments.

Instead of explicitly acknowledging every single data frame, one possible design alternative is to let other data frames piggyback acknowledgments. For example, a protocol that implements this idea has been proposed by Choudhury *et al.* [150]. Even though their protocol can provide error free communication, its performance doesn't seem to be worse than standard IEEE 802.11 in unicast mode. A similar observation has also been made by Wang *et al.* [126], who have compared the performance of IEEE 802.11 unicast and broadcast traffic. Even though their analysis shows slightly worse performance for broadcast traffic under high contention, we believe this is primarily due to the static contention window (CW) size in broadcast mode.

As a result of both studies, we come to the conclusion that there is no significant performance benefit that justifies the implementation complexity of IEEE 802.11. Although it is, of course, vital to implement the standardized behavior in order to communicate with commodity IEEE 802.11 devices, the challenge is whether this can be also accomplished with low-cost SDRs.

It should be noted, however, that the separation between data transfer and MAC functions of our architecture requires additional efforts to implement explicit acknowledgments. Consider the following example in which two STA, A and B, have data to send to each other. Imagine a situation in which the MAC of STA A has already dequeued the next data PDU destined to STA B. However, because the channel is currently occupied by another STA, STA A backs off its transmission to avoid a collision. During the backoff period, STA A successfully receives a data PDU from STA B. Recall that when receiving a data frame from another STA, an acknowledgment would be normally sent to the transmitting STA immediately. In our split architecture, however, this requires an additional mechanism to cancel the ongoing data transmission inside the MAC from within the data transfer protocol. This additional mechanism, however, would not be required if data transfer and MAC functions were implemented in a single component. Even though this renders the implementation of a monolithic protocol on the basis of our component-based architecture more complex, we believe that the overall advantages of the SoC outweigh their disadvantages.

5.3.2.5 Experimental Performance Evaluation

The last section has discussed some of the consequences that affect the implementation complexity of a IEEE 802.11 link layer. Specifically, we have found that a pure software implementation of a CSMA-MAC cannot meet the timing requirements of IEEE 802.11, alone for the reason that such an implementation would have too large of a delay between RF front end and host computer. But what performance should we expect from a pure host implementation of a CSMA-MAC? It is the main goal of this section to answer this question.

In doing so, we will employ the radio configuration depicted in Figure 5.9. We will assess the performance for a single transmitter/receiver pair (unidirectional throughput) in both AM and UM. Then, we study the performance in a multiuser scenario with up to four STAs. First, we describe the analytical model that is used to compare the experimental results with.

Table 5.5: *IEEE 802.11 MAC/PHY protocol parameters. All values in* µs*. Reproduced from [51].*

Standard	Slot time	SIFS	DIFS
IEEE 802.11a	9	16	34
IEEE 802.11b	20	10	50
IEEE 802.11g	9	10	28
IEEE 802.11ac (5 GHz)	9	16	34

CSMA Performance Model

The performance of a CSMA-MAC that follows the rules of IEEE 802.11 depends on various parameters that impact the frame transmission procedure. Among those parameters are slot time and SIFS, both of which have already been mentioned in Section 2.5.1.

The IEEE 802.11 specification describes slot time as a hardware-dependent parameter that includes all PHY and MAC layer delays, including the CCA time, the transceiver switch-over time, and the MAC processing time. For each PHY configuration, the standard defines individual slot time and SIFS values. Table 5.5 shows the default protocol parameters for different IEEE 802.11 standards. Independently from the PHY layer, DIFS is calculated by the following equation:

$$\text{DIFS} = 2 \cdot \text{slot time} + \text{SIFS}\,. \tag{5.6}$$

As shown by Heusse *et al.* [151], the total transmission time T of a data frame comprises the transmission time of the data frame itself, $t_{tr} = L/R$, where L is the length of the frame in bits and R the data rate, a constant overhead t_{ov}, plus the time spent during the contention phase, if multiple STAs (N) are active in the network:

$$T = t_{tr} + t_{ov} + t_{cont}(N)\,. \tag{5.7}$$

The constant overhead is defined as:

$$t_{ov} = \begin{cases} \text{DIFS} + t_{pr} + \text{SIFS} + t_{pr} + t_{ack} & \text{in AM (unicast transmission)} \\ \text{DIFS} + t_{pr} & \text{in UM (broadcast transmission).} \end{cases} \tag{5.8}$$

For unicast transmissions, the overhead comprises twice the preamble and header transmission time t_{pr}, i.e., for data and acknowledgment PDUs, the SIFS period, and the transmission time of the acknowledgment t_{ack}. As no acknowledgments are sent in broadcast mode, the overhead only includes the DIFS period and the preamble and header transmission time for sending the data frame [126]. Note that a random waiting period after each transmission is always needed, also in case only a single host transmits continuously. This mandatory feature, sometimes also referred to as *post-backoff* [152], is necessary in oder to prevent a single host from capturing the entire channel. The actual waiting time is uniformly distributed within $[0, CW_{min}]$ and, on average, is $\frac{CW_{min}}{2}$. Therefore, in case of only one active transmitter, i.e., $N = 1$, $t_{cont}(N)$ may be approximated as:

$$t_{cont}(N = 1) = \text{slot time} \cdot \frac{CW_{min}}{2}\,. \tag{5.9}$$

Figure 5.12 illustrates a unicast transmission of a single frame using the CSMA-MAC (without correct time scaling). The efficiency of the protocol can be calculated as the transmission time of the data frame divided by the total transmission time:

$$\eta = \frac{t_{tr}}{T}. \tag{5.10}$$

The resulting throughput TP is:

$$\mathrm{TP} = \eta \cdot R. \tag{5.11}$$

We will employ this model in the following in order to compare it against the results of our experimental implementation.

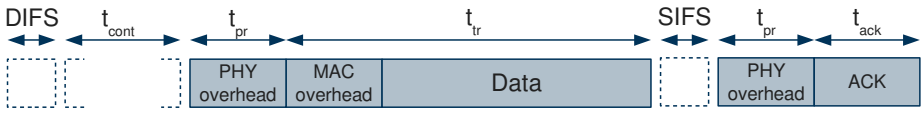


Figure 5.12: *Illustration of a unicast frame transmission in IEEE 802.11 (based on [151]).*

SoftCsma Protocol Parameters

In order to calculate the expected throughput using *SoftCsma*, it is first necessary to determine the PHY and MAC processing delays of our software implementation. For doing so, we have used a method that is described in detail in [6]. This method employs a so-called blocker device that occupies the channel for a manually adjustable amount of time. The device under test is configured such that it transmits a modulated frame immediately after turning off the blocker. The slot time may be approximated by measuring the delay between the instant at which the channel state turns from busy to idle and the time at which the SDR actually accesses the channel. It is also possible to determine SIFS using a similar measurement setup. For measuring the delay, we have used the real-time spectrum analyzer.

Table 5.6 shows the detailed results after twenty repetitions of the experiment. Due to the relatively small number of samples and unknown population standard deviation σ, we have used the t-distribution to calculate the 95% confidence interval (CI). The negative impact caused by non-deterministic processing delays can be clearly observed. Understandably, both slot time and SIFS are much higher compared to the IEEE 802.11 standard. Table 5.7 compares the most important protocol parameters of both systems. It can be observed that slot time, which is the most important parameter for accessing the channel, is about 21 times larger in our SDR configuration compared to IEEE 802.11b. The transmission of a 1500 B long data frame takes approximately 6.9 ms using our PHY configuration, and 1.1 ms at a data rate of 11 Mbit/s in IEEE 802.11b. Therefore, the ratio between transmission time and slot time is much larger in IEEE 802.11b. In order to achieve the same ration using our PHY layer, we would need to use a frame size of $\frac{21}{6} \cdot 1500\,B = 5250\,B$.

Even though such large frames are technically feasible, they are impractical in many applications. Therefore, we limited the frame size to a more reasonable value of 1500 B for the majority of experiments. As a consequence of this, we expect more collisions to occur and, therefore, a faster declining throughput with an increasing number of contending stations.

Table 5.6: *Experimental slot time and SIFS (in µs) after 20 repetitions.*

Parameter	Minimum	Average	Maximum	Standard deviation	95% CI
Slot time	312	429	576	67	399 - 460
SIFS	745	889	1072	89	852 - 926

Table 5.7: *Comparison of CSMA protocol parameters and scaling factors.*

Parameter	IEEE 802.11b	SoftCsma	Scaling factor
Slot time	20 µs	429 µs	~21
SIFS	10 µs	889 µs	~89
Average backoff ($CW_{min} = 15$)	150 µs	3.2 ms	~21
t_{tr} (L=1500 B)	1.1 ms (at 11 Mbit/s)	6.9 ms (at 1.8 Mbit/s)	~6

Even though the observed slot time is comparatively low (429 µs compared to 1.9 ms [35]), the observed SIFS is almost 90 times as large as in IEEE 802.11b. As has already been mentioned above, the underlying reason for the long latencies can be traced to the SDR hardware and signal processing delays on the host computer. Therefore, we expect the throughput in AM to be significantly lower than in UM. In contrast to IEEE 802.11b, we expect the throughput degradation to be more pronounced in our SDR configuration, because sending explicit acknowledgments creates higher overhead.

Unidirectional Throughput

After those basic protocol and system parameters have been obtained, we measured the actual throughput in AM and UM with only one active transmitter and receiver pair. We have used *nuttcp* (see Section 5.2) to generate UDP traffic with a constant bit rate. It is important to note that in our experiments, we did not observe different slot time and SIFS values for shorter and longer payload sizes. Therefore, it can be assumed that the generation and transmission of the acknowledgment frame primarily contributes to the overall SIFS duration. All protocol parameters are summarized in Table 5.8. Other variables, including preamble transmission time, t_{pr}, and acknowledgment transmission time, t_{ack}, have been calculated based on the PHY layer parameters shown in Table 5.3. For verification, we have also checked the calculated times using the real-time spectrum analyzer.

Figure 5.13 shows the average throughput measured at application level. All results are normalized to the maximum PHY throughput of 1.8 Mbit/s. For each frame length, we ran one measurement which lasted for 30 s. Depending on the actual frame length and whether the data was sent in AM or UM, we transmitted between 1.000 and 10.000 data frames in each run.

It can be observed that the measurements fit the analytical results for both AM and UM. The

Table 5.8: *SoftCsma protocol parameters. GDTP overhead for small networks as shown in Table 5.4. Physical layer overhead as shown in Table 5.3. All values in* µs *if not noted otherwise.*

Parameter	AM	UM
libgdtp header	25 B	13 B
Eth./IP/UDP header	42 B	
Slot time	429	
SIFS	889	
DIFS	1747	
CW_{min}	7	
t_{trans} (1500 B)	6912	
t_{ack}	3392	-
t_{pr}	320	
t_{ov}	6666	2070

offset between model and experiment can be explained by the additional overhead caused by the UDP, IP, and Ethernet headers that are needed by *nuttcp*. The difference between AM and UM mode can be explained by the increased overhead in AM, as expected. In our experimental system, t_{ov} for unicast transmissions is as high as 6.7 ms. It is dominated by the large values for SIFS and t_{ack}. Specifically, with a frame size of 1500 B, the average normalized throughput in AM is 0.45, or $0.45 \cdot 1.8$ Mbit/s $= 0.81$ Mbit/s. However, in UM, the throughput is as high as 0.63, or $0.63 \cdot 1.8$ Mbit/s $= 1.14$ Mbit/s, an increase of more than 40%. Of course, the larger the data frame, the lower the difference between AM and UM. The duration of t_{ack} can be explained by the low efficiency of the underlying PHY layer for short frames (compare Figure 5.3 in Section 5.2.2.1).

In summary, the large transmission overhead caused by the low PHY efficiency and the delay between RF front end and host computer decreases the achievable throughput for *SoftCsma* using explicit acknowledgments. Even though technically possible, in our opinion, the large transmission overhead caused by long waiting periods is not justifiable in practice.

The next paragraph studies the throughput of *SoftCsma* with multiple users. Due to the inefficiency of the explicit acknowledgment mechanism, however, we will only investigate UM traffic.

Multiuser Throughput

We now study the multiuser throughput of *SoftCsma*. Specifically, we evaluate the maximum as well as the saturation throughput with different CW lengths and a varying number of active STAs.

It is widely known that IEEE 802.11 exhibits a typical throughput pattern in multiuser scenarios [153, 126]. In particular, the throughput smoothly rises with the offered load until the *maximum throughput* is reached. After having reached its maximum, the throughput decreases until it asymptotically approaches the *saturation throughput*. The larger the number of contending STAs, the higher the throughput reduction beyond the maximum throughput. To the best of our knowledge, there is now previous work that has shown that the same holds true for host-

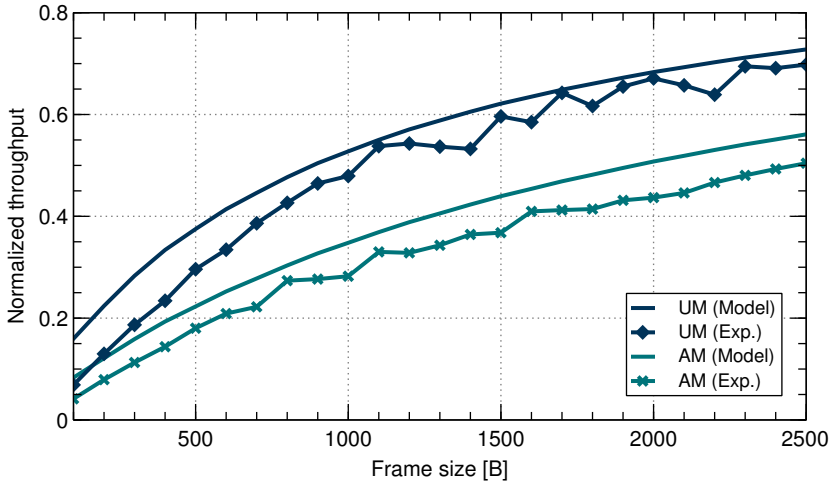


Figure 5.13: *Comparison of analytical and experimental unidirectional throughput of SoftCsma (normalized) as a function of frame size.*

based SDRs. Hence, the main purpose of the following experiment is to prove or disprove this conjecture.

We use the same setup as above but vary the number of STAs, n, between two and four. Note that each STA is a transmitter and receiver at the same time. Hence, we also have two to four active flows in the network. Furthermore, we study the impact of the CW size. Each STA produces UDP load with a constant bit rate that linearly increases in each run from 100 kbit/s to 2 Mbit/s. All results have been averaged over five runs in each case.

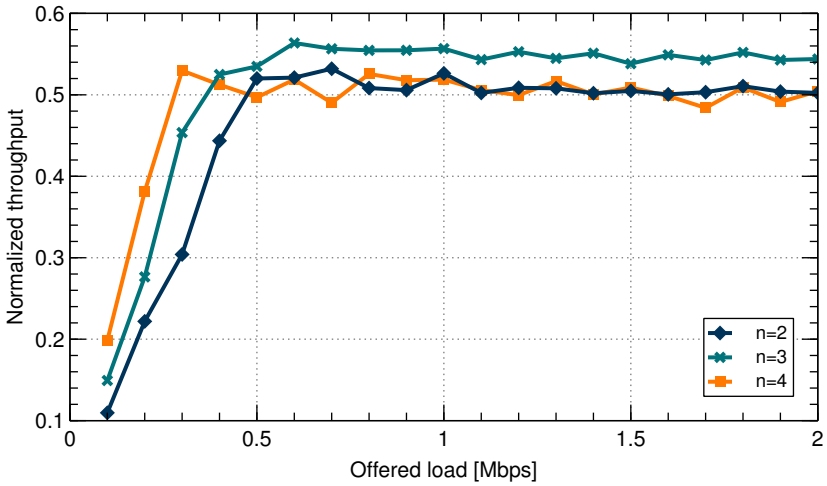


Figure 5.14: *Normalized throughput of SoftCsma in UM with a varying number of active STAs as a function of offered load ($CW = 15$).*

Figure 5.14 shows the normalized throughput as a function of offered load with a varying number of active STAs.

A number of observations can be made from the obtained results. First, in all three scenarios, the throughput linearly rises at the beginning of the experiment when the overall network load is low. After reaching its maximum, the throughput decreases and asymptotically approaches its saturation value, as has been expected. This is precisely the same behavior that can also be observed for off-the-shelf IEEE 802.11 devices [126, 154]. However, it is interesting to note that the maximum saturation throughput was achieved with $n = 3$ STAs. According to Duda, the maximum cumulative throughput in IEEE 802.11 is achieved with $n = 2$ STAs, "because the interval between transmission attempts is shorter" in this case [154]. A possible explanation for this is the interaction between the CCA mechanism and CW size which, understandably, is different in *SoftCsma*.

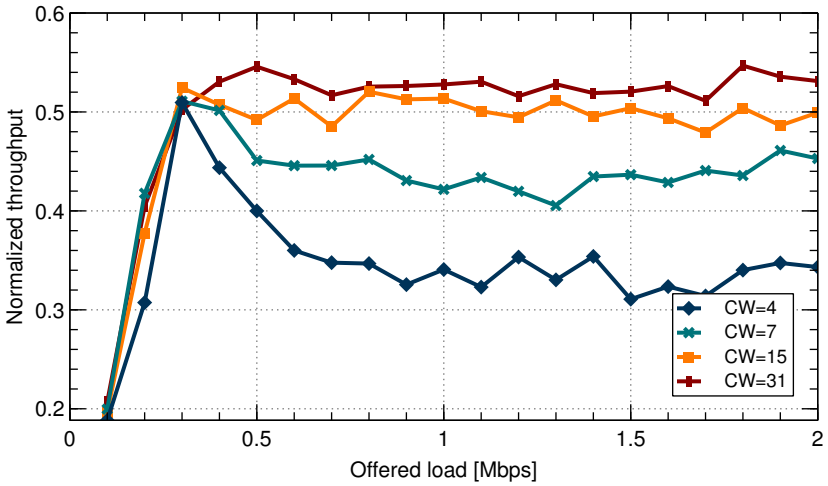


Figure 5.15: *Normalized throughput of SoftCsma in UM with a varying CW as a function of offered load ($n = 4$).*

Figure 5.15 shows the normalized throughput with four active STAs. Each plot illustrates the average computed after five repetitions using different CW lengths. One can observe the same linear rise in throughput for all four CW lengths at the beginning, when the offered load is low. However, after reaching the maximum throughput at approximately 0.55, the behavior significantly differs in each configuration.

We observed a comparatively constant throughput with larger CW lengths, even under high load conditions. With decreasing CW lengths, however, we observed a significant reduction in throughput as the contention level grew. The reason is that shorter CW lengths increase the likelihood for more than one STA to randomly pick the same backoff length. This also increases the likelihood for two nodes starting to transmit at the same time, because their backoff counter reach zero. As a consequence, more collisions occur on the shared medium which, understandably, results in lower throughput. These results not only demonstrate the functionality of the backoff algorithm, they also show that the protocol behavior of *SoftCsma* under saturation is comparable to that of IEEE 802.11.

On the basis of the obtained results, we conclude that our CSMA-MAC, even though implemented on a host-based SDR, exhibits a similar relationship between offered load and achievable throughput in UM like IEEE 802.11. Compared to commodity IEEE 802.11 hardware, however,

the throughput reduction is more pronounced with an increasing number of stations, as has been expected. Specifically, with $n = 4$ and $CW = 7$, the saturation throughput is reduced by approximately 15 % compared to the maximum throughput. We believe this can be well explained by an increasing number of collisions and the non-deterministic backoff behavior of the software realization, even though implemented with care. Furthermore, the above mentioned ratio between transmission time t_{tr} and average backoff time also influences the protocol behavior and performance.

Even though such short CW lengths may not be realistic in practice, we believe this provides an indication of the validity of these results. Furthermore, it allows better prediction of the protocol behavior if the number of STAs is further increased.

Several other researchers have also built CSMA-MACs on top of low-cost SDRs [32, 35]. However, a straightforward comparison between our results and those obtained from others is difficult because the entire PHY and link layer implementation differs significantly in all systems. It should be noted, however, that our experimental system outperforms related efforts in terms of throughput and delay.

Improving Protocol Efficiency

This section has shown that using explicit acknowledgments creates a significant overhead that lowers the efficiency of the protocol. The purpose of this paragraph is to discuss a number of possibilities for increasing the efficiency. Specifically, we propose and describe three strategies that involve both PHY and MAC layer changes.

- **Higher PHY efficiency for short frames:** Understandably, using a PHY layer which has a higher efficiency for short frames automatically also improves the efficiency of any upper layer protocol. It should be noted that this would not only decrease the transmission time for control frames (i.e., acknowledgments) but also for short data frames. For example, higher efficiency of the employed OFDM PHY layer can be achieved by tuning parameters, such as the cyclic prefix length or number of data carriers.

- **Block acknowledgments:** A further way of decreasing the relative overhead per data frame is using so-called block acknowledgments (BAs). In this mode, multiple data frames are acknowledged together using a single BA frame. For example, BAs are used in IEEE 802.11ac [37].

- **Implicit acknowledgments:** Another possibility are piggyback acknowledgments. Using this kind of implicit acknowledgment, ordinary data frames are used to also transfer the feedback needed by the error control protocol. The main idea behind this technique is to reduce the overhead needed by an explicit feedback mechanism. Piggyback acknowledgments have been explored by a number of researchers. For example, Choudhury *et al.* [150] have proposed an extension of IEEE 802.11 that uses RTS/CTS frames to transmit implicit acknowledgments. O'Sullivan *et al.* have used implicit acknowledgments to mitigate the negative effects of intra-flow collisions in multi-hop environments [155].

In order to realize implicit acknowledgments, for example, within the FLL architecture, one could implement a new scheduler in *libgdtp*. Whenever the scheduler activates a flow that has

to transmit an acknowledgment (or another data frame whose length is less than a preselected threshold), is also inspects other active flows. If any other active flow happens to have another outgoing frame in its transmit queue, this frame and the acknowledgment are combined and transmitted inside a single PHY layer resource block. If no other PDU is to be transmitted, i.e., the ready queue of all other flows is empty, an explicit acknowledgment PDU is transmitted.

Even though a rigorous evaluation of this approach for improving the efficiency of our CSMA-MAC remains future work, it is worth mentioning that we have already carried out a number of promising preliminary experiments that have confirmed the feasibility of the concept.

5.3.2.6 Summary

In this section, we have presented a random-access MAC mechanism for multiuser SDR networks that relies on the well-known CSMA principle. This MAC has been built with the intention to complement the basic functions of the GDTP without requiring any hardware modifications.

Using a prototype implementation, we have carried out a number of practical experiments to assess the unidirectional throughput of the protocol and to study the performance in multiuser scenarios. The experiments have shown that our software CSMA-MAC exhibits a similar relationship between offered load and achievable throughput like a conventional IEEE 802.11 MAC. From a scientific perspective, this is an interesting finding because CSMA-MACs are often deemed to be impossible on host-based SDRs. With certain constraints on the operating conditions and performance requirements, however, this is not the case, as our analysis has shown. The primary reasons for these constraints are the increased communication delays and the limited signal processing capabilities of host-based SDRs. In addition to that, explicit acknowledgments, which are often adopted from IEEE 802.11 for error control, are disadvantageous in high delay environments. To overcome these issues, we discussed a number of possible improvements, including implicit acknowledgments for CSMA-MACs on low-cost, host-based SDR platforms. In combination with a bare MAC implemented inside the FPGA, we believe this would provide a powerful, flexible, and cost-effective experimental basis for future research.

Regardless of the implementation-specific limitations discussed in this section, however, the performance reduction with an increasing number of contending stations is one of the main shortcomings of CSMA-MACs. As has been discussed, the main reason for that is their random access characteristic. Because STAs do not have a guaranteed transmission time, collisions may occur anytime if two STAs try to access the channel at the exact same instant of time. However, this problem can be overcome by employing a so-called reservation-based MAC. Reservation-based MACs assign each STA its own transmission time. Therefore, collisions are less likely to occur. In the next paragraph, we will explore how such a reservation-based MAC may be realized on top of the FLL architecture.

5.3.3 BasicTdma: A Hardware-assisted TDMA-MAC for Iris

This section shows how reservation-based MACs may be realized within the FLL architecture. We demonstrate how one specific MAC scheme may be replaced by another MAC, without modifying the data transfer related functions provided by another component.

For this purpose, we design and implement a basic TDMA-MAC on top of *Iris* that coordinates channel access among a set of nodes in a timely manner. The time synchronization mechanism

is thereby provided by the underlying radio hardware. In contrast to *SoftCsma* presented in the previous section, *BasicTdma* can provide better throughput characteristics in dense environments, because it assigns fixed time slots to its users. Therefore, it doesn't suffer from frame collisions. We begin with a description of the system and component architecture before explaining how the synchronization between nodes has been achieved in the practical implementation. We then analyze the throughput and compare the results against those obtained for *SoftCsma*.

5.3.3.1 Architecture and Principle of Operation

The MAC is implemented as a single component for *Iris* called *BasicTdmaTaggerComponent*. Its main purpose is to determine and enforce the actual transmission time of each frame. It implements a basic scheduling algorithm based on statically allocated fixed length slots. However, it does not provide time or time drift synchronization among nodes. If desired, such a mechanism could be implemented as an additional link layer application on top of the actual MAC. Figure 5.16 shows the high-level design of the *BasicTdma* radio for *Iris*.

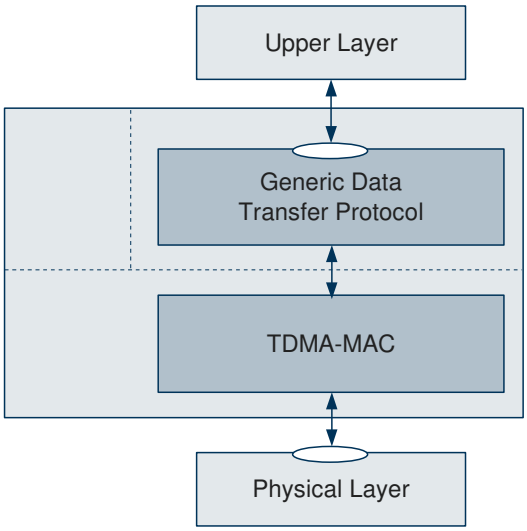


Figure 5.16: *High-level design of a TDMA radio implemented on the basis of the FLL architecture.*

In *BasicTdma*, each node i is assigned a unique identifier (ID) a_i with $i = 0, 1, \ldots, n-1$, where n is the number of nodes in the network. Moreover, each node is assigned a fixed slot within one TDMA frame f, where a_i marks the position of the slot within f. Based on the current time t_{now} and its ID, each node can precisely calculate the start of its next slot, t_{a_i}, i.e., its next transmission opportunity. First, with l_f being the length of a TDMA frame, the beginning of the next frame, t_f, can be calculated as follows:

$$t_f = \lceil \frac{t_{now}}{l_f} \rceil \cdot l_f \,. \tag{5.12}$$

Then, with l_s being the slot length, t_{a_i} is defined as:

$$t_{a_i} = t_f + (a_i \cdot l_s) \,. \tag{5.13}$$

Table 5.9: *BasicTdma protocol parameters. GDTP overhead for small networks as shown in Table 5.4. Physical layer overhead as shown in Table 5.3.*

Parameter	Value
Number of STAs	3
Ethernet/IP/UDP overhead	42 B
libgdtp overhead	13 B
Payload length	1500 B
TDMA frame length (l_f)	24 ms
TDMA slot length (l_s)	8 ms

Figure 5.17 illustrates the frame and slot structure of *BasicTdma*. The transmission time for each frame passing through the component is calculated using this algorithm. This time is then attached to the frame as meta data that is interpreted by the USRP transmitter component of *Iris* for precisely scheduling the transmission time using the timed commands capability provided by UHD. In order to provide the required time synchronization between nodes, we have used the synchronization setup described in Section 5.2.1.

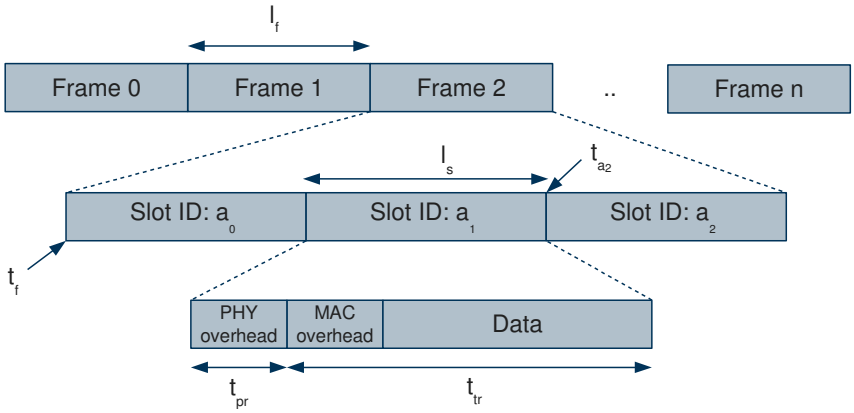


Figure 5.17: *Illustration of the frame and slot structure of BasicTDMA.*

5.3.3.2 Experimental Performance Evaluation

This paragraph studies the performance of *BasicTdma* using the same methodology as above. We will focus on UM in order to be able to compare the results against *SoftCsma*. Recall that the number of STAs is limited to $n = 3$ due to the restrictions of the time synchronization setup. The protocol parameters are listed in Table 5.9.

We have plotted the measurement results in Figure 5.18. It can be seen that the protocol exhibits reduced performance at lower traffic levels compared to CSMA. The throughput rises gradually until it reaches its maximum and stabilizes. However, in contrast to CSMA, the throughput does not degrade after reaching its maximum. This can be well explained by the absence of

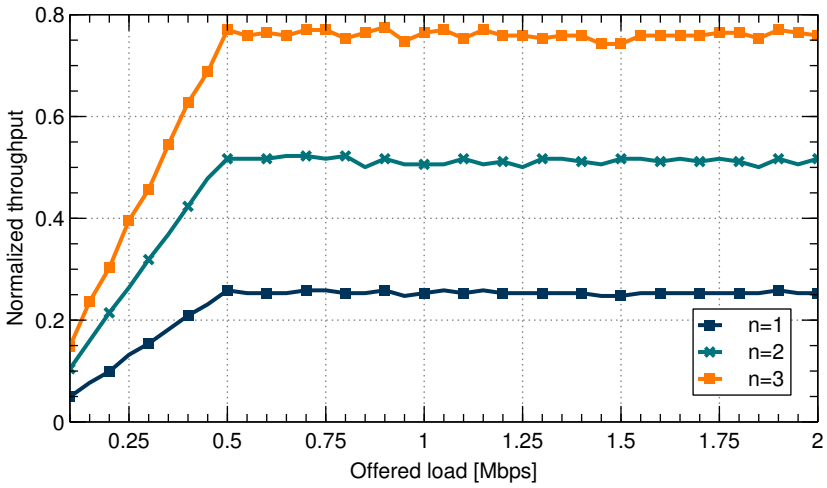


Figure 5.18: *Normalized throughput of BasicTdma in UM with a varying number of STAs as a function of offered load.*

frame collisions, which is helpful especially if the network is highly utilized. For example, with three active STAs and an offered load of 2 Mbit/s, the CSMA protocol only achieves a normalized throughput of approximately 0.55, or $0.55 \cdot 1.8$ Mbit/s $= 0.99$ Mbit/s, whereas the TDMA link layer can carry a constant load of nearly 0.78, or $0.78 \cdot 1.8$ Mbit/s $= 1.4$ Mbit/s, in the same scenario.

In contrast, if the network is only moderately used, the measurements show that a good amount of the available bandwidth is wasted because the time slots are not reallocated. Thus, the efficiency of the TDMA link layer is low if traffic density is low. Clearly, a more sophisticated protocol with dynamic slot allocation could significantly improve this situation. However, adding and removing users during runtime also generates a certain amount of overhead that needs to be considered. Furthermore, time slots that have been allocated once can't be easily reused if a node has simply no data to send.

Even though frame collisions are less likely in TDMA than in CSMA systems, transmission errors still occur, for example, due to signal propagation effects, such as fading or shadowing. Therefore, even TDMA systems should employ some kind of error control mechanism in order to cater for those. Even though we have also verified error control capabilities in TDMA operation, we do not provide detailed measurements in AM because a fair comparison is impossible at this stage. The reason for this is that frame concatenation is currently not implemented in *libgdtp*. As a consequence, each acknowledgment frame currently occupies an entire TDMA slot, resulting in poor performance at the moment. This, however, is only an implementation-specific issue that will be solved in the future.

5.3.3.3 Summary

In this section, we have described the realization of a simple reservation-based MAC mechanism for *Iris*. Following the SoC principle, we have implemented the TDMA-MAC as a separate component underneath the GDTP implementation. This allowed us to reuse the GDTP that has already

been part of the CSMA radio discussed earlier.

In contrast to CSMA, the TDMA-based link layer has shown to provide constantly high throughput, even in dense network environments. This is due to the nature of the reservation-based access scheme that assigns fixed transmission times to each node and, thereby, reduces frame collisions.

5.3.4 Summary

This section described our efforts in transferring the idea of a GDTP with decoupled MAC mechanism into reality. We have designed and implemented *libgdtp* as an open-source library that can be used for both simulations and real-world deployments. We then evaluated its performance in terms of protocol overhead and proved its suitability for small- and large-scale networks.

In an attempt to demonstrate the applicability of the concept for SDR networking, we then implemented and evaluated a random-access as well as a reservation-based MAC for *Iris*. The main contributions of this section are threefold. First, the benefits of variable-length encoding were shown. Specifically, it was demonstrated that the protocol overhead for small payload lengths could be reduced significantly, by up to 23%, compared to fixed-length encoding. Second, it was shown that our CSMA-MAC, even though fully implemented in software on a host-based SDR, exhibited the typical IEEE 802.11 DCF throughput pattern for broadcast traffic. This allowed us to build a small practically working SDR network without requiring any additional hardware or FPGA modifications. Lastly, it was demonstrated that the separation of media-independent and media-dependent functions of the link layer allows to transparently use the GDTP for error control purposes on top any MAC scheme, even on top of a TDMA-MAC.

However, the obtained results have also revealed some of the weak aspects of static, general purpose protocols. It has been shown that the achievable performance is not always satisfactory, especially when operating at the edge of the indented application scenario. But couldn't an adaptive protocol provide better throughput in such cases? In an attempt to provide an answer to this question, we will now explore the capabilities of the FLL architecture in more detail and investigate how adaptive protocols can help improving the performance in fixed-spectrum networks.

5.4 Adaptation in Fixed Spectrum Networks: A Load-adaptive Link Layer

In the previous section, we have studied and compared the performance of wireless communication systems with static, general-purpose link layer configurations. We have shown that the CSMA-MAC achieved favorable results if the overall load of the network was relatively low. On the other hand, our TDMA-MAC has shown to provide constantly high throughput, especially when node and traffic density was high.

In this section, we will explore how the benefits of both MAC schemes can be combined in a single system. Specifically, we will design, implement, and evaluate a load-adaptive link layer that allows switching between both schemes based on the observed traffic density. In this section, we will once again demonstrate the reuse of link layer components, one of the key design goals

and advantages of the proposed FLL architecture. The content of this section has been published in the following paper:

- André Puschmann and Andreas Mitschele-Thiel. Implementation and Evaluation of a Flexible, Load-Adaptive Link Layer Protocol. In *20th Annual International Conference on Mobile Computing and Networking (MobiCom), 9th ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH)*, pages 73–80, Maui, HI, USA, September 2014

5.4.1 System and Component Architecture

In order to benefit from the advantages of a CSMA-MAC in periods with low load and from the advantages of a TDMA-MAC during periods with high traffic density, we have complemented the existing link layer architecture by two more blocks:

- **Handover Negotiation Protocol:** The handover negotiation protocol (HNP) is a distributed protocol responsible for negotiating the MAC handover target.
- **Handover Controller:** The controller coordinates the interaction between HNP, the MAC components, and the remaining radio components.

In combination with our existing, non-adaptive MACs, the HNP and the handover controller facilitate switching between different MAC schemes during runtime. During operation, a MAC handover may be requested by any node within a set of nodes of a collision domain based on the analysis of performance metrics collected by each node individually. Specifically, we exploit the frame error rate (FER) as an indicator for detecting high network contention. In contrast, we use the channel busy fraction (CBF) as a PHY layer indicator for detecting low network load. Both switching metrics will be explained below.

In order to find a common MAC among a set of nodes, a distributed consensus finding algorithm is required. For this purpose, we have implemented the HNP as a LLA on top of the GDTP. The second newly developed component is a link layer controller that coordinates the handover procedure and the interaction among the HNP and the rest of the architecture. The core components of our load-adaptive link layer protocol are depicted in Figure 5.19. We will now explain each component and the MAC switching metrics in detail.

5.4.2 MAC Switching Metrics

The actual switching metric of the protocol depends on the activated MAC scheme.

Frame Error Rate

If the protocol runs in CSMA mode, we utilize FER as handover metric. The FER can be determined at the receiver by analyzing the sequence numbers of incoming frames. For this purpose, *libgdtp* provides a mechanism that allows external components to query the current FER or to periodically report the FER to the link layer controller. The actual FER is calculated locally over all incoming connections.

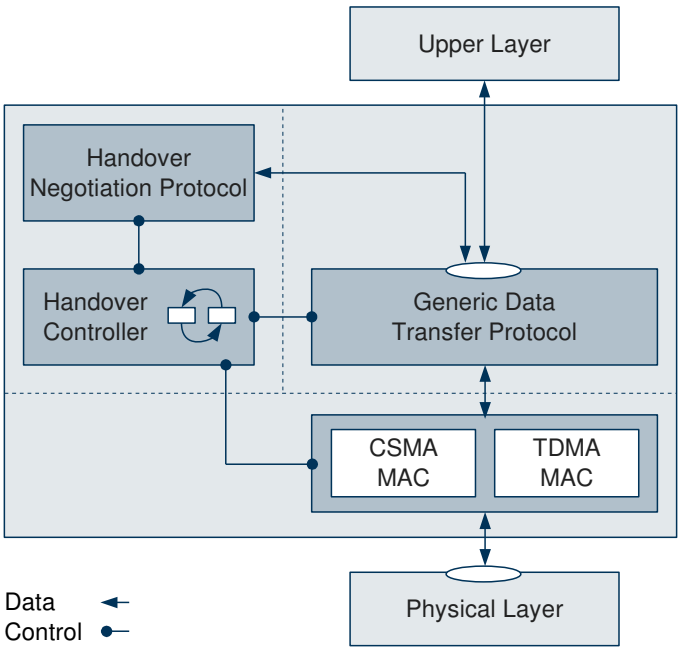


Figure 5.19: *High-level design of the load-adaptive link layer protocol implemented on the basis of the FLL architecture.*

Channel Busy Fraction

If the protocols runs in TDMA mode, evaluating the FER for the purpose of detecting a possible handover situation to CSMA is meaningless. This is because the FER of an ideal TDMA system is zero[5].

A simple approach to overcome this problem is speculatively switching back to TDMA after a certain amount of time, betting that network load has reduced again. This approach, for example, has been taken by *AMAC* [115]. Understandably, the main drawback of this method is that the overall system performance is reduced again in case the load conditions have not changed. With each unneeded handover to CSMA, we effectively force a second handover back to TDMA. In addition to that, the selection of a proper timeout value that provides a good balance between responsiveness of the system and number of unneeded handovers is also challenging.

Therefore, we propose to employ the channel busy fraction (CBF) as a performance metric for controlling the protocol switching. CBF is defined as the fraction of time in which the wireless channel is occupied due to ongoing transmissions. This metric has been used in previous research, for example in rate adaptation algorithms. It has shown to be a useful indicator for the available bandwidth of the wireless channel [156].

We compute the CBF over a time period as the arithmetic mean of n single measurements of the received power using the following equation:

$$\mathrm{CBF} = \frac{1}{n}\sum_{i=1}^{n} cb_i. \tag{5.14}$$

[5] We note that in practice, frames may still get lost in a TDMA system due to other effects, such as fading or shadowing.

The resulting CBF is a real number between $[0,1]$. For determining the received power of the channel in practice, we exploit the same ED algorithm that has been used for CCA of the CSMA-MAC. Again, we assume the channel to be busy if the power level $y(t)$ is equal to or exceeds the predefined threshold $\gamma(k)$, and assume it to be idle if the power is below the given threshold:

$$cb_i = \begin{cases} 1 & y(t) \geq \gamma(k) \\ 0 & \text{otherwise.} \end{cases} \tag{5.15}$$

In the experimental system, CBF estimation is implemented inside a new PHY component for *Iris* called *CbfEstimatorComponent*. The component receives the measured power as a real numbered value from the *EnergyDetectorComponent* on its input port. The detection threshold (in dBm) as well as the window length (in ms), i.e., the observation period, are configurable component parameters. The component uses a periodic timer (set to the window length) for continuously producing CBF estimates. After each period, the component sends an event containing the computed CBF to the link layer controller.

5.4.3 Operation of the MAC Handover Protocol

Regardless of the target protocol and the evaluation metric, the handover algorithm runs through four consecutive phases described by the protocol state machine illustrated in Figure 5.21.

For a better understanding of the protocol operation, consider a CSMA-based network in which an increasing number of STAs content for the channel. As more and more STAs trying to access the shared channel, more collisions occur and, therefore, the FER increases. Figure 5.20 illustrates the development of the FER curve over the course of the MAC handover process, as observed from a single STA. The four phases of this process can be summarized as follows:

- **Idle:** In idle mode, the system continuously receives FER and CBF measurements and compares them against the configured thresholds. Depending on the currently active MAC protocol, either FER or CBF values are considered. If the measured value exceeds the threshold, the state machine switches into the evaluation state.

- **Evaluation:** When entering the evaluation phase, the evaluation timer is started. The hysteresis, during which all samples must fulfill the threshold criterion, is needed to avoid ping pong effects. If these requirements are met, an event is triggered and the state machine switches to the negotiation state.

- **Negotiation:** If a node is in the negotiation phase, it has *locally* decided that it wants to perform a protocol handover. However, its neighbor nodes may not have reached this state yet and, thus, need to be informed about the intended handover. Therefore, the handover negotiation protocol (see below) is started. If the neighbor nodes accept the handover request, the protocol enters the reconfiguration phase.

- **Reconfiguration:** In this phase, the handover command is executed on all nodes and the requested MAC switch is performed.

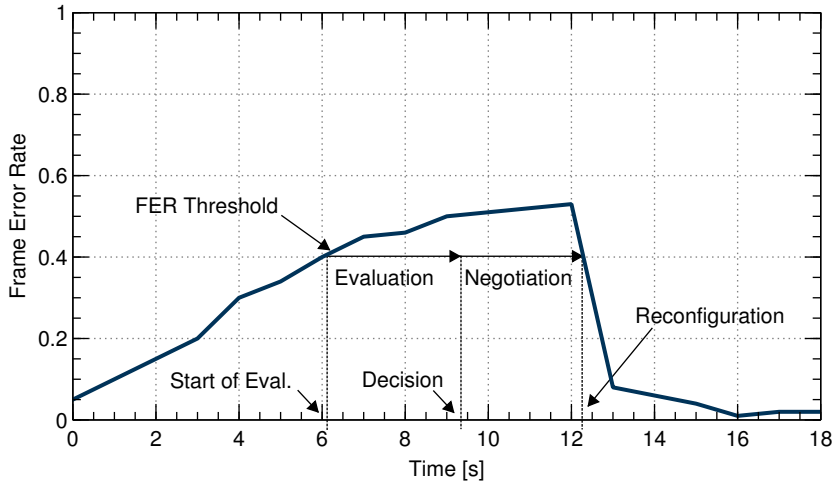


Figure 5.20: *Visualization of FER as switching metric between CSMA and TDMA operation.*

5.4.4 Handover Negotiation Protocol

During the design of the FLL architecture, we have mentioned the necessity of a feature negotiation mechanism. This mechanism would allow to change a certain feature of a communication system during operation in a reliable manner. This is, the mechanism guarantees that the participants of the communication have successfully agreed upon the value of a feature.

The handover negotiation protocol (HNP) described in this section provides such a mechanism to its clients. The HNP is a three-way handshake protocol to coordinate the MAC handover procedure between an initiator and one or more neighbor nodes. We assume nodes to be completely reliable, i.e., not providing any erroneous information. For information exchange, the HNP uses the GDTP in AM to guarantee reliable transmission between protocol instances. Figure 5.22 depicts the message sequence chart (MSC) of a typical conversation between two nodes.

The HNP is activated when entering the negotiation state of the controller state machine. After activation, the HNP requests a list of all active neighbors from the GDTP. A *«HO_SYN»* message is then generated and transmitted by the initiator protocol instance to all of them.

If a neighbor node does not agree on the proposed handover, it sends a *«HO_SYN_NAK»* back to the initiator. Depending on the acceptance policy, the initiator may discard the handover as soon as at least one neighbor disagrees. In this case, no further messages are exchanged. Figure 5.22a depicts the corresponding MSC.

However, if a neighbor agrees on the proposed handover, it sends a *«HO_SYN_ACK»* message back to the initiator. If all neighbors agreed upon the requested handover, the initiator responds with a *«HO_ACK»* that triggers the actual protocol handover on all participating nodes at the same time. In practice, there may be variations between the actual handover instances. This could be accommodated for by inserting time stamps into the handover frames.

If the initiator doesn't receive a valid *«HO_SYN_ACK»* or *«HO_SYN_NAK»* from all neighbors within a specific amount of time, the requested handover is also discarded. For example, this could happen if a node fails, moves away or is turned off during communication.

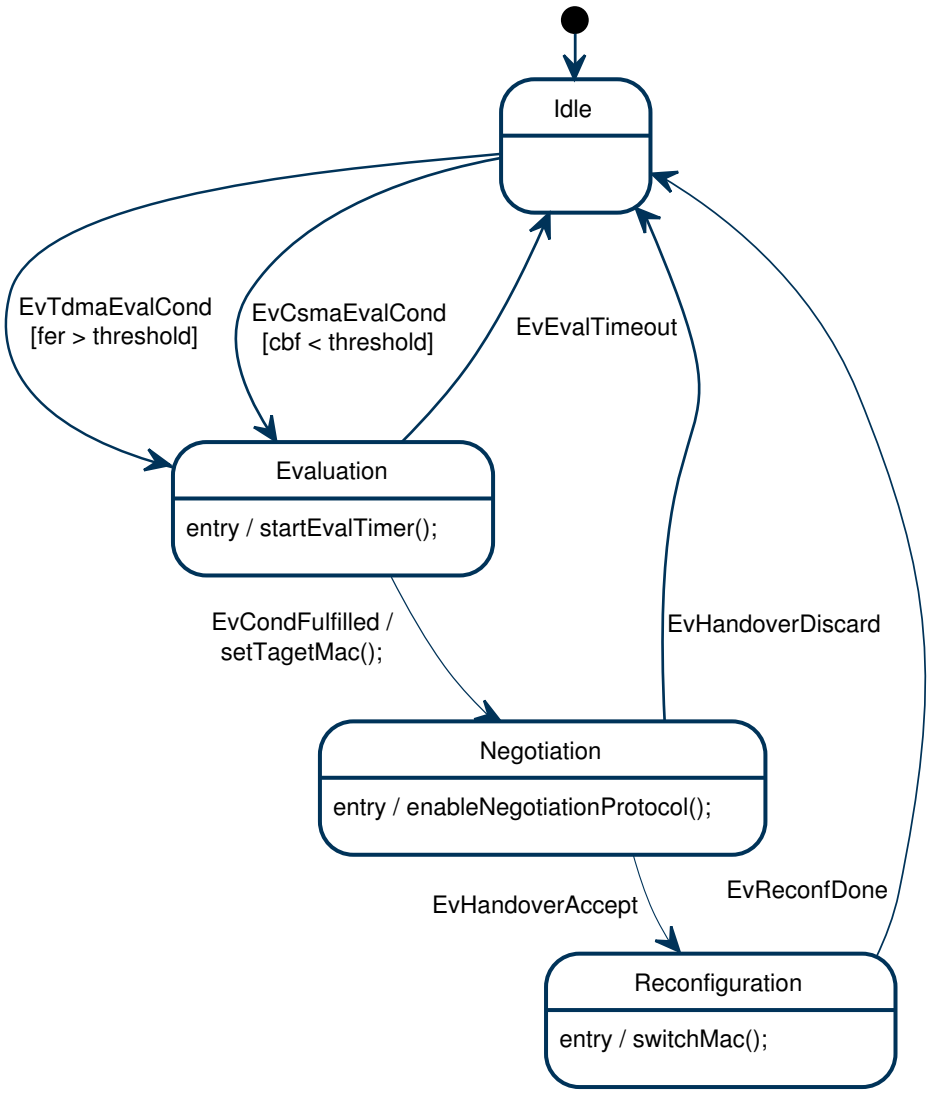


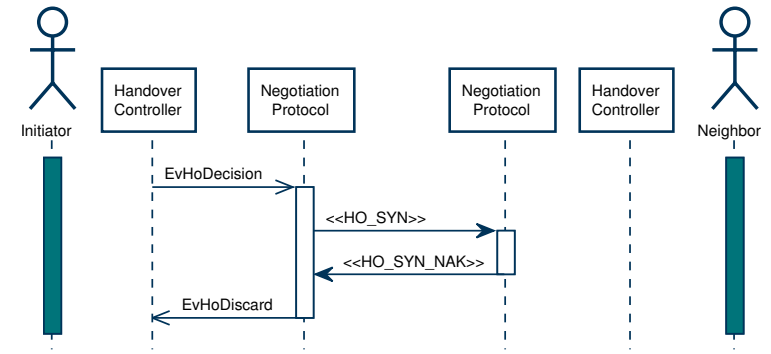
Figure 5.21: *MAC handover controller state machine.*

5.4.5 Experimental Performance Evaluation

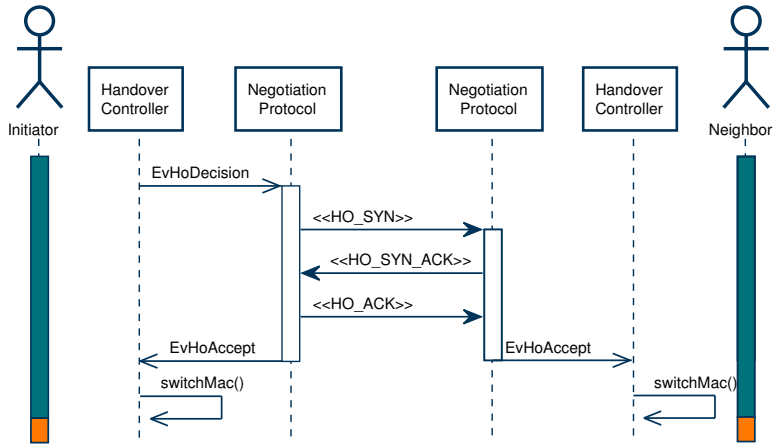
In order to evaluate the performance of the load-adaptive link layer protocol for *Iris* in terms of data throughput, we have created an experimental setup that aims at reproducing a realistic network scenario with varying traffic load. The setup comprises three identically equipped SDR nodes transmitting constant bit rate (CBR) data between one another.

The experiment lasts 90 s in total. Each flow, however, has its own starting time, duration, and load. Figure 5.23 illustrates the time line of the experiment and shows the configuration of each flow in the network scenario.

Figure 5.24 shows the throughput of each flow as well as the combined throughput. The plot is the result of one particular run of the experiment with the FER and CBF threshold set to 0.2 and 0.4, respectively. The handover evaluation time was 3 s. We started the experiment on an



(a) *Unsuccesful MAC handover.*



(b) *Successful MAC handover.*

Figure 5.22: *MSC of the handover negotiation protocol between the handover initiator and a neighbor node. The colored bar represents the MAC that is currently active.*

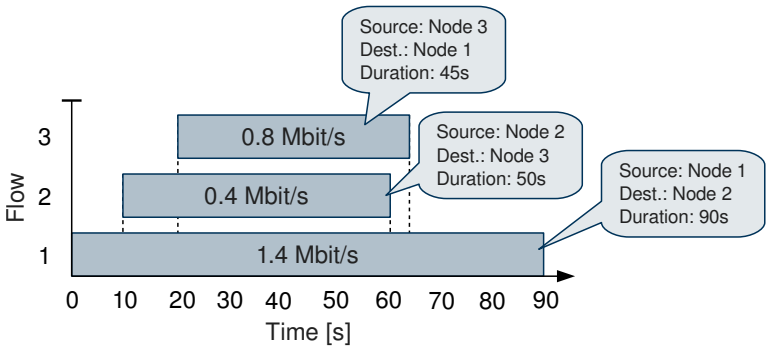


Figure 5.23: *Detailed flow configuration of the network scenario.*

idle system in CSMA mode. We can see that the throughput was high for the first 10 s with only one active flow. However, as soon as the second flow was started, the individual as well as the overall throughput declined, as collisions started to occur. At this point, the FER was not high enough yet to exceed the configured FER threshold. Therefore, the link layer continued to run in

Table 5.10: *Comparison of average throughput and relative improvement of the load-adaptive link layer compared to the static, non adaptive baseline protocols.*

Protocol	Average norm. throughput	Average throughput	Difference to TDMA	Difference to CSMA
TDMA	0.52	0.93 Mbit/s	-	-18.8%
CSMA	0.64	1.14 Mbit/s	+23%	-
Load-adaptive	0.69	1.23 Mbit/s	+32%	+7.2%

CSMA mode.

However, after 20 s, right after the third user began to transmit, the observed throughput dropped even further, and the FER exceeded its configured threshold. Note that if the FER observed by a node is above the threshold for the entire evaluation time, it triggers a handover decision and activates the HNP. The HNP, in turn, sent a *«HO_SYN»* frame to all neighbors with TDMA set as target MAC. Similar to the example MSC depicted in Figure 5.22b, the neighbors agreed on the request and sent a *«HO_SYN_ACK»* frame back to the initiator. Finally, the initiator transmitted a *«HO_ACK»* frame to all neighbors to trigger the actual handover.

In this particular example, the CSMA to TDMA handover took place after approximately 24 s. From that moment on, the protocol exhibited its TDMA characteristic and served all flows with an equal share of the available time, as can be seen in the middle section of Figure 5.24.

After termination of flow two and three after 60 s and 65 s, respectively, a second handover from TDMA back to CSMA took place. This second handover occurred because the utilization of the channel fell below the configured CBF threshold.

Figure 5.25 compares all three link layer protocols in the networking scenario. As we can see from the plot, the load-adaptive protocol performs as expected, offering the benefits of CSMA at times with low or only moderate use of the channel, and the benefits of a TDMA-MAC at times with a highly loaded channel. The results also show that a speculative switch from TDMA back to CSMA after a certain amount of time is not required in the presented approach, which may reduce the total number of handovers. Table 5.10 compares the average throughput of all three link layer configurations and also shows the relative difference between them. From the results, it can be seen that the load-adaptive link layer achieved a 7% increase in average throughput compared to the static CSMA-MAC. Compared to the TDMA-MAC, the throughput could be increased by 32%. Note, however, that the actual benefits are highly load- and scenario-specific.

5.4.6 Summary

The previous section has demonstrated the applicability of the FLL architecture for developing general purpose link layer protocols, for example, by exploiting a conventional CSMA-MAC. However, while static protocols may provide reasonable performance in certain environmental conditions, they usually fail to properly address issues they were not designed for in the first place. This problem may be overcome by either designing a single protocol that is capable to also support those corner cases, for example, through protocol adaptation. Another approach to overcome the

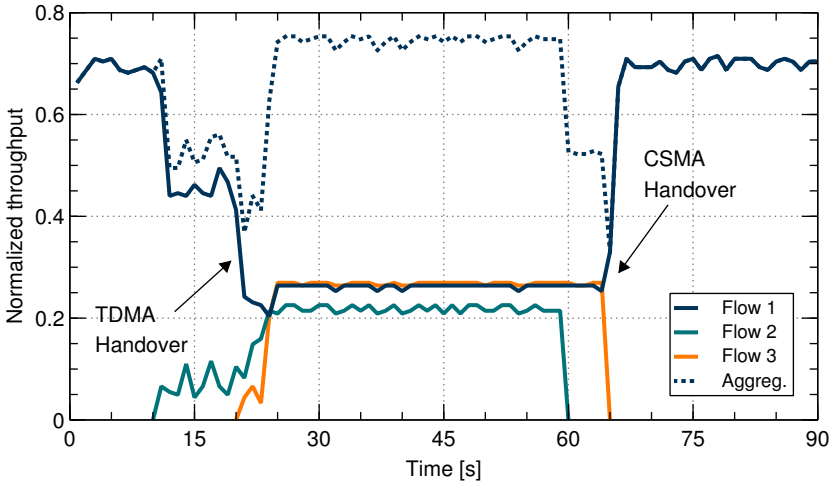


Figure 5.24: *Per-flow throughput of the load-adaptive protocol in the network scenario.*

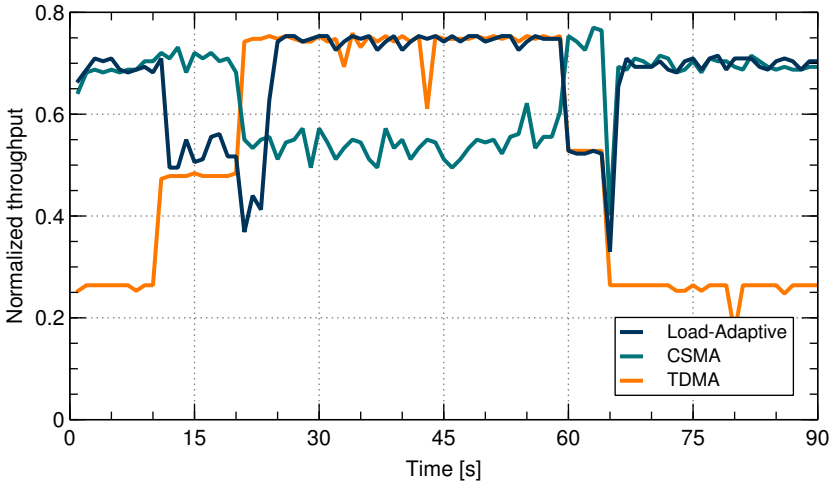


Figure 5.25: *Comparison of the combined throughput of all protocol variants in the network scenario.*

issue is designing an adaptive protocol framework that is able to combine the advantages of two or more standalone protocols in a flexible manner.

In this section, we have explored the latter path by designing such a protocol framework. The key advantage of our load-adaptive link layer implementation is that its main components, i.e., the GDTP as well as the MACs, could be reused from existing implementations without any modification. This step significantly reduced the development time of the entire link layer architecture. In future, it will also allow to easily integrate further protocol evolutions.

To facilitate the evolutionary development of protocols in general, the FLL architecture postulates the separation of concerns concept. In this specific case, we have complemented two existing, independent MAC implementations by another protocol, the handover negotiation protocol (HNP). The HNP allows to transparently switch between MAC schemes such that the most

appropriate solution can be used with respect to the current network conditions. In our specific scenario, this increased the average throughput by 7%, compared to the static CSMA-MAC, and by 32%, compared to the static TDMA-MAC.

Even though the obtained results demonstrate the potential benefit of adaptive protocols in order to address the challenges of FWCS, the achievable performance gain is still limited, because the capacity of a single radio channel is limited. In order to further increase the achievable throughput of FWCS, new schemes for accessing the available radio spectrum may be needed. Those new methods may no longer rely on a policy that only allocates fixed resources to a certain communication provider or technology. In contrast, radio users may be allowed to opportunistically exploit available spectrum in a dynamic and flexible manner. The FLL architecture has been specifically designed with DSA systems in mind. The next section demonstrates the applicability of the proposed concept in such environments.

5.5 Adaptation in Dynamic Spectrum Access Networks

In the previous section, we have seen how intelligent and adaptive protocols may help to increase data throughput in FWCS. However, due to the rapid increase in wireless communication, we have reached a situation in which some frequency bands are completely overloaded. This includes, for example, unlicensed spectrum in the 2.4 GHz ISM band. Therefore, given any channel with a fixed bandwidth, the achievable performance gains are ultimately bounded, regardless of possible advances in PHY and link layer processing. In contrary, large portions of the licensed spectrum are only moderately used or even completely unused. This inefficient and unbalanced assignment of spectral resources gave rise to the development of DSA systems. The main idea behind DSA is to relieve heavily occupied frequency bands by offloading parts of the traffic to other, less occupied bands in an opportunistic manner. As we have seen in Chapter 2, DSA is considered to be one of the key technology components of FWCS. One question that immediately comes to mind is that of whether the FLL architecture is capable to support such systems.

Providing an answer to this question is the main goal of this section. For doing so, we implement multiple state-of-the-art spectrum mobility and link establishment protocols and compare their performance through computer simulations and practical experiments. We address decentralized as well as centralized, i.e., infrastructure-based, deployments. Our results demonstrate that the envisioned architecture is well suited for implementing even complex protocols for DSA environments in a flexible manner.

We continue our journey by studying spectrum mobility first, even though link establishment happens prior to that in practice. We then expand our study and also explore the problem of autonomous link establishment. We use this order because link establishment is more restrictive than spectrum mobility in terms of the underlying assumptions. First of all, we provide a detailed description of the network scenario.

5.5.1 Network Scenario

For a better understanding of the considered network scenario and its challenges, recall the concept of DSA networks introduced in Section 2.2.2. Figure 5.26 illustrate a typical overlay DSA network that comprises a set of shared communication resources as well as a number of PU and

SU devices. All nodes are within the communication range of each other, i.e., they reside in a single collision domain. Table 5.11 shows a summary of the parameters and scenario assumptions which will be explained in detail below.

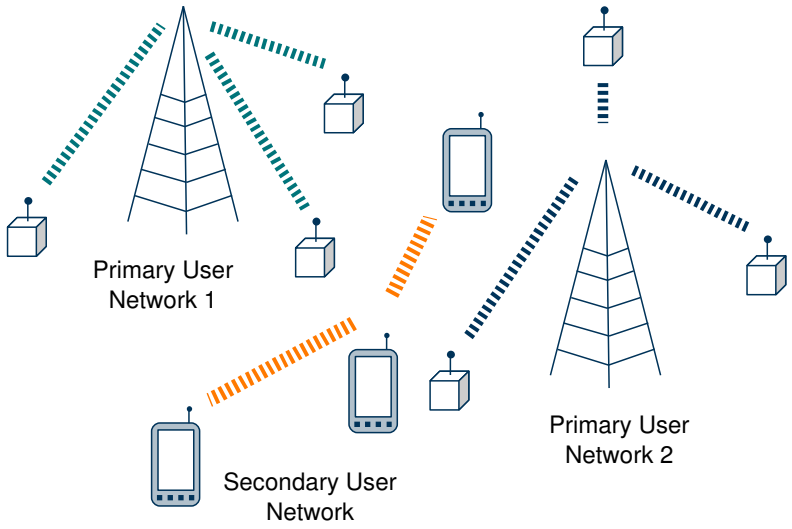


Figure 5.26: *Simplified sketch of an overlay DSA network comprising two PU networks, each with several terminal devices operating on two distinct frequencies, and a SU network opportunistically reusing the frequency band not occupied by the PU networks.*

Communication Resource Assumptions

In the considered network scenario, no radio spectrum will be allocated to a single device or a group of devices exclusively. Furthermore, it is not guaranteed that a single channel is permanently unoccupied. As a consequence, the use of a static control channel (CC) for coordination is infeasible. In the considered scenario, the radio spectrum is divided in a total of $M (M \geq 1)$ orthogonal, i.e., non-overlapping, channels. It is also important to note that we are assuming an overlay scenario in which all nodes transmit at the same power. Furthermore, we do not explicitly differentiate between licensed and unlicensed frequencies. Needless to say, the SU, i.e., our experimental FWCS, may only occupy the spectrum while the PU is not actively using it. After the PU appears in a channel occupied by the SU, the SU is required to vacate that band immediately.

We further assume that all channels are labeled uniquely from $1, 2, \ldots, M$ and that all users know the labels. Therefore, the overall set of available channels is $C = \{1, 2, \ldots, M\}$ with $|C| = M$.

Primary User Assumptions

We assume L independently operating PU networks. Each of those networks may occupy one of the available channels at all times. However, PUs may also cooperate with each other. Therefore, different PU networks may also share a single channel temporarily. PU activity is modeled through specific busy/idle patterns that are further described below.

Secondary Users Assumptions

For our spectrum mobility analysis, we consider a centralized SU network with pre-established links. Specifically, we assume a cluster of K SUs, each equipped with a single reconfigurable radio transceiver that is tuned to the current operating channel of the cluster. A coordinator (or cluster head) is assumed to be elected among all nodes [157]. Hence, the remaining $K-1$ nodes act as so-called "followers".

We further assume a symmetric environment in which all K nodes are within the transmission range of each other (single collision domain). They also observe the same set of available channels, thus $C = C_1 = C_2 = \cdots = C_K$. The coordinator is equipped with a second reconfigurable radio receiver that is used for determining the state of the remaining channels. For this purpose, the coordinator periodically hops over all channels, listens for PU activity on a particular channel, and reports the results to the link layer controller. As a consequence, all nodes require a mechanism to classify the available spectrum. This mechanism should be capable to distinguish between occupied and unoccupied frequency bands. If a certain portion of the spectrum is occupied, the mechanism is further required to differentiate between PU activity and SU activity.

Table 5.11: *Summary of simulation parameters and scenario assumptions.*

Description	Spectrum mobility	Link establishment
Spectrum access model	overlay, single collision domain	
Channel model	symmetric	symmetric/asymmetric
Number of channels (M)	1-10	1-100
Number of PUs (L)	1-10	-
PU activity	on/off with 5 load patterns	-
Number of SUs (K)	≥ 2	2

5.5.2 Coordinator-based Spectrum Mobility

From a system architecture point of view, even the brief description of the network scenario poses a number of significant question that need to be answered. These questions include:

- Which channels are available for communication?
- How can the appearance of a PU be detected?
- If the current channel is no longer available, which channel can be used instead?
- How can one make sure that all nodes are aware of the new channel?

Research on spectrum mobility in DSA networks has addressed most of these issues and has also suggested possible solutions, yet the practical realization of these proposals is limited [66, 158, 111, 63, 112]. Even though all these questions may be discussed isolated from each other, the particular challenge for building a practically working system is answering *all* of them.

In order to address the above mentioned challenges, we will now describe the design, development, and evaluation of a coordinator-based spectrum mobility protocol that implements a hybrid handover strategy. In other words, the protocol negotiates a so-called backup channel (BC) among a set of nodes in a proactive manner. However, the BC is not actively used until a PU actually occupies the current operating channel of the cluster (reactive channel handover).

This protocol has been realized as an example to showcase the applicability and practicability of the FLL architecture for DSA networks. Understandably, the dimension of the practical experiments needed to be limited due to resource constraints. Therefore, we have also implemented a basic simulation tool to assess the performance of the protocol at a larger scale. In the following, we describe how we approached the problem of accurately modeling PU activity. We then describe the practical implementation, after which the simulative analysis is explained.

5.5.2.1 Primary User Activity Model

To abstract from the actual activity of a real PU network, researchers usually employ models that try to mimic the behavior of a real PU. The majority of such models typically differentiates between an active and an idle PU. One example is the simple two-state Markov chain (MC) activity model [159, 160]. In this model, the busy state indicates the period in which a PU is active in a particular frequency band, while the idle state indicates that the PU is not active. In the two-state discrete-time Markov chain (DTMC) model, the PU transfers between these two states with certain probabilities, thus generating an alternating pattern of busy and idle periods (see Figure 5.27).

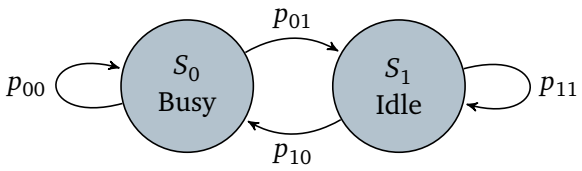


Figure 5.27: *Two-state Markov chain for modeling primary user busy/idle periods (reproduced from [160]).*

In recent years, a great number of studies aiming at characterizing the activity of PUs has been carried out [161, 160, 162]. It has been shown that a stationary discrete-time MC is not able to reproduce the statistical properties of busy and idle periods of real systems [160]. As a result of this, continuous-time MC models, in which busy and idle periods are exponentially distributed, have been proposed. Although this approach is widely used [67, 15], it doesn't reproduce PU behavior adequately [160].

Depending on the PU technology, e.g., IEEE 802.11, GSM, TETRA, and the PU load, i.e. low, medium, high, different stochastic distributions have shown to provide a good fit to the given scenarios. For example, Geirhofer *et al.* [161] has shown that a hyper-Erlang distribution provides a good approximation of empirical data for modeling the busy/idle durations of a IEEE 802.11 network transmitting both data and voice traffic.

A more generic model, covering a wider range of technologies and spectrum bands, including GSM, DECT, TETRA, and IEEE 802.11, has been proposed by López-Benítez and Casadevall [162]. In their work, a continuous-time semi MC (CTSMC) is used. A CTSMC is capable to let the

Table 5.12: *Parameters of the Generalized Pareto Distribution (in seconds) for modeling busy/idle periods of PUs (reproduced from [162]).*

Load	DC	Busy periods			Idle periods		
		μ	σ	ξ	μ	σ	ξ
Very low	0.09	3.5150	1.6960	0.0284	3.6100	38.3633	0.2125
Low	0.29	3.5150	2.6240	0.1884	3.5780	10.9356	0.1784
Medium	0.51	3.5150	5.1483	0.1978	3.5160	4.6583	0.2156
High	0.71	3.5470	10.7968	0.1929	3.5310	2.6272	0.2119
Very high	0.93	3.5940	52.8611	0.2377	3.5160	1.6609	0.0068

busy and idle periods follow any arbitrary distribution. Further, it is shown that the generalized Pareto distribution (GPD) provides a good approximation for fitting the busy and idle periods of real-world measured data. As a consequence, we have used the model of López-Benítez and Casadevall in our analysis to model the PU activity pattern. Specifically, we will use a set of reference parameters given in [162] that have been obtained from empirical measurements for five different PU load patterns (very low, low, medium, high, very high). The parameters of the GPD for all five levels, specified by three parameters, i.e., location μ, scale σ, and shape ξ, are shown in Table 5.12.

In principle, our PU activity also consists of repeating busy and idle periods, similar to those generated by the model shown in Figure 5.27. However, instead of selecting static transition probabilities between busy and idle states, our model uses the parameters provided in [162] to "reproduce the statistical properties of busy and idle period lengths of real channels" [160]. In each step, the PU remains in the busy state for a period with mean $\mathrm{E}\langle t_{busy}\rangle$ and then in the idle state for a period with mean $\mathrm{E}\langle t_{idle}\rangle$. The load, or duty cycle (DC) Ψ, as it is called in [162], may be calculated as:

$$\Psi = \frac{\mathrm{E}\langle t_{busy}\rangle}{\mathrm{E}\langle t_{idle}\rangle + \mathrm{E}\langle t_{busy}\rangle}\,. \tag{5.16}$$

5.5.2.2 System and Component Architecture

In order to provide spectrum mobility capabilities to our communication system, we again have to extend our current link layer design by a number of components. Clearly, we base our new design on the existing GDTP and MAC implementation. However, we need to complement them with three new blocks:

- **Coordinator Mobility Protocol:** The coordinator mobility protocol (CMP) is responsible for selecting and communicating an appropriate handover target channel.
- **PU Detector:** This component detects and signals the activity of the PU.
- **Mobility Controller:** The mobility controller coordinates the interaction between mobility protocol and the PU detector. It also controls the periodic sensing of all available channels.

It understands the concept of a BC that is used as handover target channel and also carries out the radio reconfiguration.

Figure 5.28 shows the high-level design of a link layer configuration with spectrum mobility capabilities.

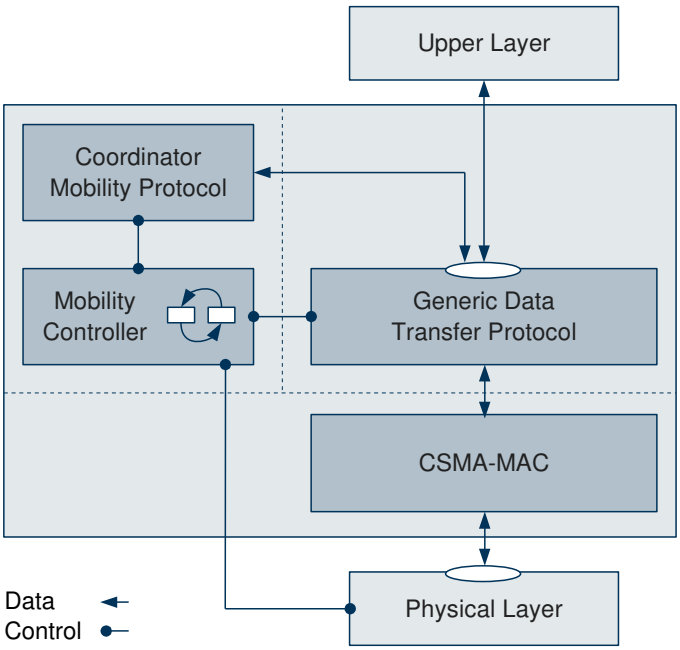


Figure 5.28: *High-level design of a link layer with spectrum mobility capabilities.*

5.5.2.3 Protocol Operation

Recall that our mobility solution is a hybrid protocol. Consequently, the protocol runs through two independent phases:

- **Proactive Target Channel Selection:** In order to quickly respond to the sudden appearance of a PU, the available radio spectrum needs to be monitored continuously and an appropriate target channel needs to be chosen. The coordinator is mainly involved in this phase of the protocol, followers are only passive participants.
- **Reactive Channel Handover:** After an appropriate BC has been chosen, the channel handover itself, triggered through PU activity event, is controlled by each node individually. This allows to reduce the handover delay and guarantees that no further interaction between coordinator and follower is required.

Figure 5.29 depicts a simplified flow graph of an *Iris* radio implementing the coordinator-based protocol. Note the generic part on the right side of the figure that is the same for coordinator and follower nodes. The left part of the figure depicts the components for proactive spectrum sensing, which are only needed by the coordinator. Figure 5.30 shows the corresponding MSCs of the communication between coordinator and follower nodes.

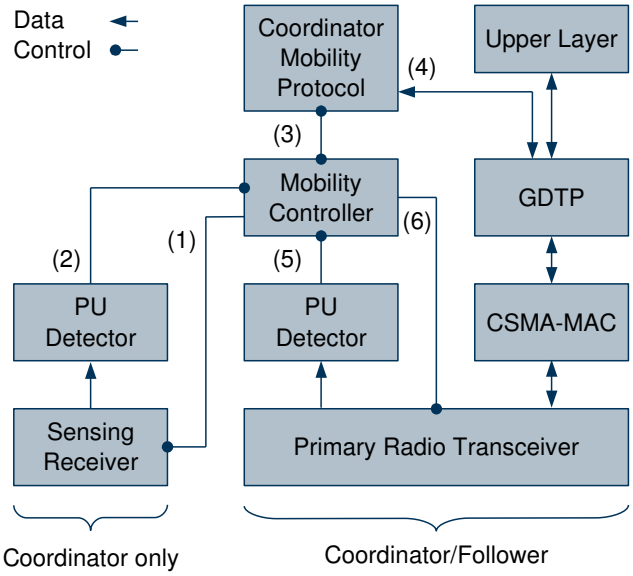
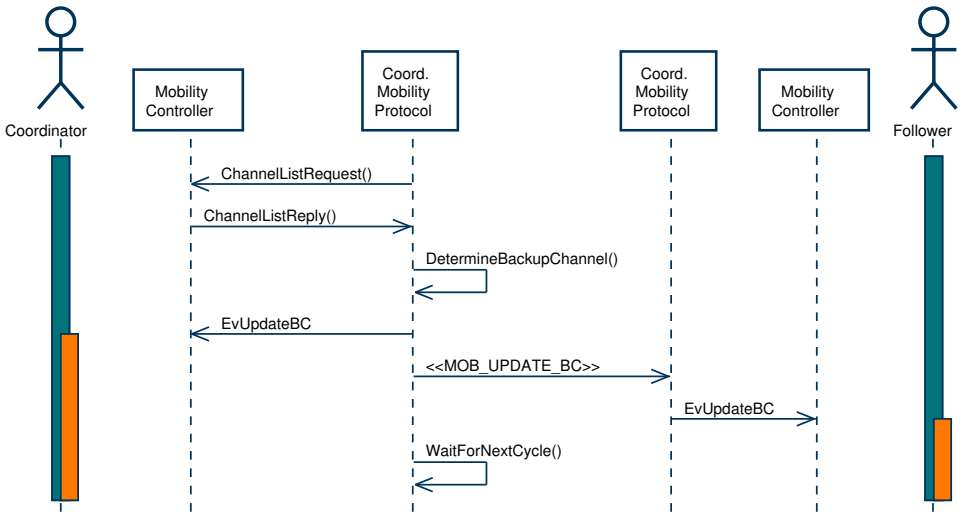


Figure 5.29: *Flow graph (simplified) of the Iris implementation of the spectrum mobility protocol. The left part of the radio is needed at the coordinator node only.*

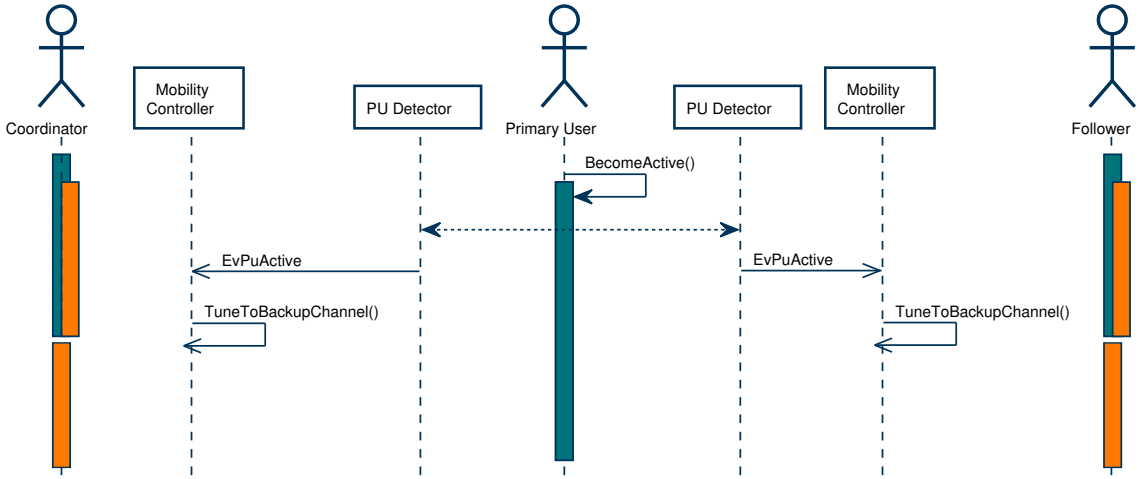
The *MobilityController* monitors the channel activity by sweeping through the set of available channels. For this purpose, one part of the controller HSM periodically reconfigures the sensing receiver (see tag (1) in Figure 5.29) and listens for possible PU transmissions for a certain amount of time, see tag (2). The state of each channel, i.e., busy or idle, is stored inside the generic information storage element of the controller for later use. Note that the spectrum monitoring sub-phase is fully asynchronous.

The actual target channel is selected by the CMP. The protocol instance running on the coordinator node periodically queries the *MobilityController* to obtain the set of currently available channels, see tag (3) in Figure 5.29. The CMP then iterates through this list and selects the first channel that has been reported to be idle as the new BC. If none of the channels is vacant, the new BC is randomly selected. The coordinator then informs its own mobility controller about the new BC, see also tag (3). However, more importantly, the BC is also broadcast to all followers which then inform their spectrum mobility controller accordingly, see tag (4). Because the CMP is a soft-state protocol [163, 164], periodic refresh messages are needed to keep the shared state among the coordinator and followers consistent.

As has been mentioned above, the actual channel handover itself is controlled by each node individually, in order to keep the handover delay as short as possible. For this purpose, the receiver chain of the primary radio transceiver of each node includes a PU detection component that continuously monitors the actual operating channel. If the component detects a PU transmission, it immediately informs the mobility controller by triggering a PU activity event, see tag (5) in Figure 5.29. As a consequence, the mobility controller promptly reconfigures the radio in order to not interfere with the PU, see tag (6).



(a) *Proactive target channel selection.*



(b) *Reactive channel handover.*

Figure 5.30: *MSCs of the coordinator-based spectrum mobility protocol. The colored activity bar represents the channel of the radio. The primary channel is green. Orange has been selected as backup channel. After the PU appears on the green channel, both SUs switch to the orange channel.*

5.5.2.4 Practical Primary User Detection

Fast and accurate detection of PU activity is a key requirement for DSA systems in order to avoid interfering with them. Possible solutions include the use of beacon signals and databases that store when and where PU system will be active [65, 165]. Another possibility is to use spectrum sensing [64] to autonomously detect PU activity. During our studies, we have primarily focused on spectrum sensing and explored energy detection (ED) as well as feature detection based approaches [166].

In our experiments, we have found that energy detection has the lowest overhead but suf-

fers from reduced detection probability and increased probability of false alarm. This may yield longer detection delays and, thus, unacceptable channel handover times, i.e., above the specified channel detection time (CDT) [167].

In order to get more reliable PU detection results, we have explored different feature detection approaches. In general, feature detection aims at capturing a distinct property of a transmitted signal, such as a pilot or a synchronization field; something that is known a priori. In our experiments, we exploit the fact that we have full control over the PU signal and, thus, are able to dictate:

- the bandwidth of the PU signal,
- the type of modulation,
- the number and allocation of carriers, and
- the structure of the header symbols.

Even though this may be a requirement that can't be always met in practice, the existence of a similar mechanism can still be assumed.

For practically implementing this strategy within *Iris*, we have chosen to use a similar PHY layer configuration for both PU and SU.

In fact, the only difference between PU and SU transmissions is a unique 8 B long identifier that we embedded into the OFDM header symbols of the PU signal. This identifier allows to reliably distinguish between PU and SU transmissions without requiring any complex signal processing. This detection mechanism has been implemented as an *Iris* component called *FeatureDetectorComponent*.

5.5.2.5 Performance Evaluation

The goal of this section is to quantify the potential performance improvements of spectrum mobility in the considered communication scenario.

Methodology

As part of this thesis, we have fully implemented the coordinator-based spectrum mobility protocol as a prototype based on our FLL architecture. This prototype has been used as a vehicle to successfully demonstrate and verify the practical feasibility at various conferences and technical events, for example, at the *10th International Symposium on Wireless Communication Systems (ISWCS)*, held in Ilmenau, Germany in August 2013 [8]. However, even though we believe practical experiments are invaluable for evaluating novel communication protocols, they also have a number of disadvantages, such as their high complexity and limitation with respect to reproducibility. Furthermore, they are often restricted to small scale studies.

As a consequence, we have designed and implemented a Python-based computer simulation that allows to assess the efficiency of spectrum mobility strategies more easily, time efficient, and in a reproducible manner. The simulation tool is called *MobilitySim* [168]. *MobilitySim* allows to study spectrum mobility with various PU activity patterns, with varying number of available

channels, and varying handover times. Therefore, this section will be primarily concerned with computer simulations for evaluation purposes.

However, for obtaining meaningful simulation results, it is also essential to use realistic parameters. As a consequence, we will first exploit our practical implementation for determining realistic handover times, that are then used for the simulation.

Primary User Assumptions

In order to reproduce a realistic PU behavior, we employ the model proposed by López-Benítez and Casadevall that has been described above. In addition to different PU activity levels, we also differentiate between static and dynamic PU channel selection. With static channel selection, each of the L PUs is assigned one of the available channels at the beginning of the simulation. The PU remains on that channel for the entire simulation run. Channels are also not shared among PUs. However, in the dynamic case, each PU selects its operating channel at the beginning of each busy period with uniform probability $p = \frac{1}{M}$. In this scenario, multiple PU may even share the same channel[6] during their busy periods, depending on the outcome of the random channel selection. Note that this may have an impact on the average channel occupation. For example, during one busy period, two PUs may select the same channel for communication. This may render a channel unusable for a longer time period, because the busy periods of both PUs are not aligned. At the same time, however, another channel may be less occupied, thus creating a longer transmit opportunity for SUs.

For our simulations, we assume the SUs to have a mechanism to reliably detect PU activity on all channels. In other words, we assume perfect sensing. In order to estimate the achievable performance of our coordinator-based spectrum mobility protocol, we have to consider two distinct cases, depending on the number of available communication channels. In case of only one available channel, i.e., $M = 1$, the communication channel will have to be shared among PUs and SUs. According to our access policy, this implies that the SU can only use the channel if and only if the PU is not using the channel itself. Note that this is exactly the no handover strategy that has been discussed in Section 2.4.3.

In case of more than one available channel, i.e., $M > 1$, the PU may select the current SU operating channel, thus causing the SU to perform a spectrum handover. With proactive target channel selection, we can assume that SUs have already negotiated a backup channel (with negligible overhead during normal communication). However, each spectrum handover takes a certain amount of time, denoted as $t_{handover}$, during which the SUs can't communicate. Should the selected channel be occupied after the handover, the SU network remains on the channel and backs off its transmission until the PU disappears.

Performance Metric

As has been mentioned above, the primary goal of the experiments is to evaluate the efficiency of spectrum mobility in an overlay DSA scenario. With this in mind, consider a data flow between two SUs that transfers data at an average rate R, e.g., a large file of size S. Depending on the employed SU technology, R may vary between a few kbit/s and several Mbit/s

[6] This may require some kind of coordination among PUs, which is, however, not uncommon, e.g., in enterprise IEEE 802.11 deployments.

The time needed to transfer the file in an ideal network, t_{ideal}, without any interference, neither due to SU nor PU activity, may be calculated as:

$$t_{ideal} = \frac{S}{R}. \tag{5.17}$$

However, in the considered DSA scenario, the time t_{dsa} needed to transfer the same file of size S may be longer. This is primarily because of PU activity and possible spectrum handovers during communication. Clearly, the more handovers are needed during communication, the more time is needed to transfer the file. Thus, the time needed to transfer the same file in the DSA scenario may be calculated as follows:

$$t_{dsa} = t_{ideal} + \sum_{i=1}^{n} (t_{handover} + t_{wait_i}), \tag{5.18}$$

where n is the total number of handovers during communication, $t_{handover}$ the constant handover overhead, and t_{wait_i} the individual waiting time for each handover i. Note that t_{wait_i} is zero if at least one idle channel exists and non-zero otherwise.

Rather than evaluating spectrum mobility by comparing the transfer times or average data rate directly with each other, we use the efficiency η, measured in percent, as a performance metric:

$$\eta = \frac{t_{ideal}}{t_{dsa}}. \tag{5.19}$$

Note that solely using transfer time as a performance metric has advantages as well as disadvantages. One advantage is that it allows one to also apply the obtained results to other SU technologies. In other words, it provides a technology-agnostic means for evaluating the potential performance gains of DSA-capable networks. In contrary, however, this model may also hide certain effects that only come to light when using a specific SU technology. For example, the employed model assumes that all spectral holes are exploitable, regardless of how long they last. In practice, however, this may not be the case because certain technologies may require a minimum time span for any successful transmission.

Experimental Handover Time

As has been stated above, the experimental results explained in this section are mainly for the sake of determining a practically meaningful value for $t_{handover}$. For not underestimating the spectrum sensing duration and accuracy, a simple ED-based algorithm has been used on purpose, knowing well that its performance may not be optimal in a more realistic RF environment. Note that this is in contrast to our practical demonstrations that mostly relied on feature detection.

The spectrum sensing procedure is carried out periodically with varying intervals. At the beginning of each interval, the radio receives $n = 4096$ I/Q samples[7], over which the energy level is averaged. During the experiment, one node continuously transmits data to another node. The actual handover time is measured at the receiver by calculating the throughput with a resolution

[7] Note that the actual number of samples is not fixed. More or less samples could be used for more accurate or faster detection, respectively.

Table 5.13: *Experimental handover time (in seconds) with different sensing intervals after 25 repetitions (reproduced from [7]).*

Interval	Minimum	Average	Maximum	Standard deviation	95% CI
1.00	0.40	1.31	3.40	0.86	0.95 - 1.66
0.75	0.40	0.89	1.80	0.47	0.68 - 1.08
0.50	0.30	0.74	1.50	0.35	0.59 - 0.88

of 100 ms. The sensing interval has been varied between 500 ms and 1 s. For each interval, the experiments have been repeated 25 times.

Table 5.13 shows the results of the experiment. We observed the minimum handover time to be almost the same for all sensing intervals. Assuming that the PU activity could be detected successfully shortly after it has been activated, 300 ms can be seen as the minimum handover time for our experimental system. While the minimum value is almost the same for all interval values, the average value clearly decreases with shorter sensing intervals, as expected. Regardless of the interval length, the maximum handover time was always larger than twice the actual interval. This can be explained by performance of the ED algorithm. Due to the non-zero probability of misdetection at either the transmitter or receiver, two or even three cycles may be needed to successfully detect the presence of the PU. A more accurate detection algorithm could clearly improve the worst-case handover time in this case. Even with the longest sensing interval, the average handover time measured was approximately 1.3 s, which is well below the maximum CDT of 2 s defined in IEEE 802.22. To be on the safe side for our simulations, we round up the largest average value to 1.5 s.

In the following, we first study the impact of the number of channels and PUs using this value as a realistic handover time approximation. We then study the impact of varying handover times separately.

Simulation Results: Impact of Number of Channels and Primary Users

Figure 5.31a shows the SU efficiency in presence of PUs with static channel selection and various activity patterns. The efficiency is shown as a function of the number of channels. The actual number of PUs was set to match the number of available channels, i.e., $M = L$. The simulated time was one day. We can make a number of observations from the obtained results.

First, if $M = L = 1$, i.e., only one PU and one pair of SUs have to share a single channel, the efficiency is exactly $1 - \Psi$ for all PU activity patterns. This can be explained by the fact that SUs can only access the channel during the idle periods of the PU.

Second, with an increasing number of available channels and PUs, the efficiency of the SU transmission increases for all traffic patterns except for very high PU loads. This can be explained by the fact that the actual time slots that the SU may use for transmissions are extremely short in this scenarios, even if multiple channels are available. Thus, the time spent waiting for a PU to vacate a channel again is significantly larger in this case. In all other cases, SUs can benefit from non-overlapping PU idle periods that allow SUs to transmit for an overall longer amount of time.

Specifically, in the case of medium loaded PUs and six available channels, the efficiency can be improved by over 30%, from 49% to 80%, compared to a single channel network. With very high PU loads, however, it is likely that all remaining channels are also occupied by PUs, if SUs have to vacate their current operating channel. Therefore, we can't observe an improvement in this case, even with a growing number of channels, as can be seen in Figure 5.31a.

The results of the same scenario but with dynamic PU channel selection are plotted in Figure 5.31b. We can observe that even under high PU loads, SUs can benefit from a growing amount of available channels. This is because each PU is not exclusively bound to a single channel. Instead, PUs hop over the available channels and may share them among themselves, therefore creating more opportunities for SUs. Generally, it can be said that the higher the PU load, the less efficient the SU transmission. However, one exception are highly loaded PUs with dynamic channel selection. For example, with $M = 10$, the efficiency was even higher than under low PU loads. This can again be explained by the long busy periods in that scenario. For instance, if at least two PUs select to transmit on the same channel, it is more likely that they do so for a relatively long amount of time, thus creating a transmit opportunity for SUs in other parts of the spectrum. From Figure 5.31b, it can also be seen that SUs can't benefit from more than 6 channels.

Simulation Results: Impact of Handover Time

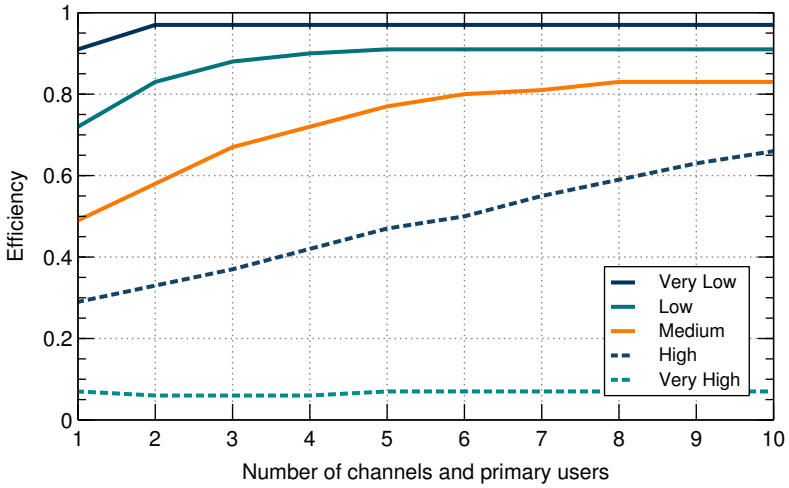
Figure 5.32 illustrates the efficiency as a function of handover time for PUs with static channel selection under various loads. We have varied the handover time between 100 ms and 2 s[8]. The number of channels as well as the number of PUs was set to five. Generally, it can be observed that the efficiency decreases with increasing handover delay, as expected. For medium loaded PUs, the reduction is more pronounced because in this scenario, handovers happen more frequently than in all other scenarios.

5.5.2.6 Summary of Spectrum Mobility

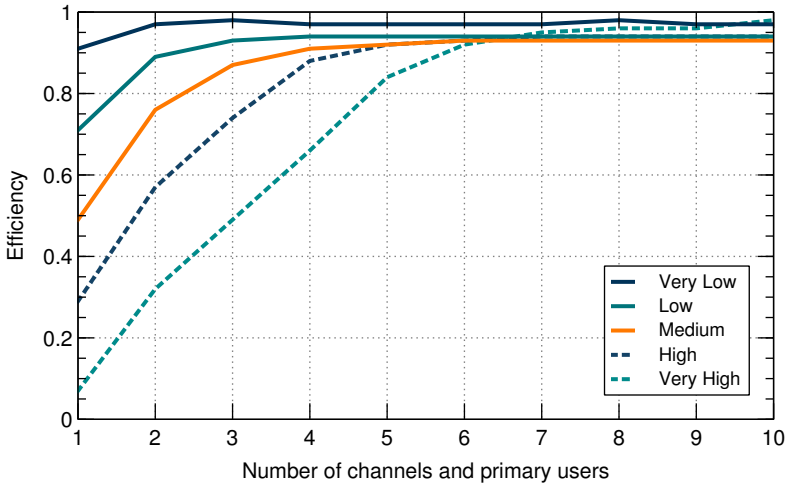
This section has addressed the issue of persistently maintaining an opportunistic communication link between two SUs while guaranteeing the protection of a PU network in a DSA environment. In order to solve this issue in a real-world communication system, various important questions need to be answered, such as how to detect the presence of a PU transmission. We have presented our approach to those in this section. More importantly, however, we have practically demonstrated the suitability and applicability of the FLL architecture for building such systems that actually work.

Admittedly, our system model made a number of assumption that cannot always be met in real-world systems. Specifically, the coordinator mobility protocol assumes a cluster of nodes that have already established a common communication channel among each other. This assumption, however is rather unrealistic, considering a distributed mobile ad-hoc network with possibly thousands of communication channels and no predetermined network configuration. Consequently, there needs to be a mechanism that allows nodes to establish links among themselves on their own. The purpose of the next section is to address exactly this issue.

[8]Note that the CDT within the IEEE 802.22 standard is required to be ≤ 2 s [167, 166].



(a) *PUs with static channel selection.*



(b) *PUs with dynamic channel selection.*

Figure 5.31: *Simulated efficiency of spectrum mobility as a function of available channels and PUs* ($t_{ideal} = 24$ h, $t_{handover} = 1.5$ s, $M = L$).

5.5.3 Practical Link Establishment

Establishing links among network nodes, logically and physically, is essential in any communication network before information can be exchanged. However, link establishment is not considered to be a tremendous issue in traditional systems, either because a centralized controller may assist nodes in setting up links among each other, or because the overall number of potential communication channels is quite small. In contrast, link establishment may become a difficult problem to solve in a distributed mobile ad-hoc network with possibly thousands of communication channels and no predetermined network configuration.

In this section, we develop the mechanisms for our FLL architecture to assist link establishment in FWCS. For this purpose, we design an algorithm-agnostic handshaking protocol for the FLL

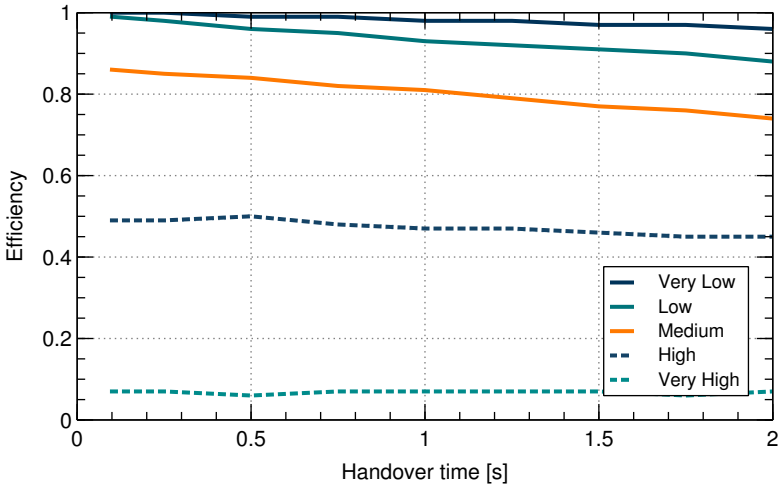


Figure 5.32: *Simulated efficiency of spectrum mobility as a function of handover time for PUs with static channel selection under various loads ($t_{ideal} = 24$ h, $M = L = 5$).*

architecture and *Iris* that is used to demonstrate the integration of a link establishment mechanism into existing protocols.

Moreover, we review representative algorithms and evaluate them under different configurations. We again rely mainly on computer simulations because practical experiments with a large number of nodes and radio channels are extremely difficult to carry out. Therefore, we have developed a protocol simulator in Python that allows to precisely control all simulation parameters. The problem we are specifically interested in is to quantify the difference in time needed to establish links in centralized networks compared to decentralized networks.

We begin by revising and extending the underlying network model. We then briefly describe the establishment procedure in CH systems using a single radio transceiver before presenting the newly developed *Iris* component. The end of the section evaluates the practical experiments as well as our computer simulations.

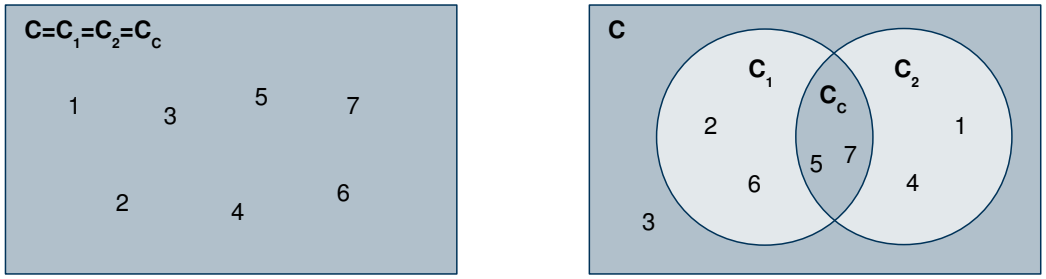
5.5.3.1 Network Scenario Adjustment

We relax some of the requirements posed on the network scenario considered in this section. In a realistic network, the set of available channels may differ among nodes. This is because nodes may be geographically separated from each other and, thus, their spectrum sensing module may provide different results. Therefore, each user i for $i \in \{1, 2, \ldots, K\}$ maintains an individual list of channels C_i.

Users are said to operate in an *asymmetric model* if their channels differ. If all users observe the same set of available channels, the model is referred to as *symmetric* [52]. In general, we assume G ($G \geq 1$) commonly available channels. The set of commonly available channels is denoted as C_C, with $|C_C| = G$. In the symmetric case, $C = C_C$. In the asymmetric case, this is usually not the case.

To give an example, consider a radio environment with seven channels $C = \{1, 2, 3, 4, 5, 6, 7\}$ and two users, radio 1 and radio 2, with channel sets C_1 and C_2 and $C_1, C_2 \subseteq C$. In the symmetric

case, both users have access to all available channels, thus $C_1 = C_1 = C$ and $G = 7$. In the asymmetric case, $C_1 \neq C_2$. Let's assume $C_1 = \{2,5,6,7\}$ and $C_2 = \{1,4,5,7\}$. In this case, both users could only rendezvous in $C_C = C_1 \cap C_2 = \{5,7\}$. Regardless of the model, rendezvous is only possible if $C_1 \cap C_2 \neq \emptyset$. Figure 5.34 depicts an example of both symmetric and asymmetric channel distributions. Moreover, and without loss of generality, we set $K = 2$, i.e., we assume a two user scenario[9].



(a) *In the symmetric model.* **(b)** *In the asymmetric model.*

Figure 5.33: *Example channel distribution between two users.*

5.5.3.2 Protocol Operation

Regardless of the network scenario, i.e., centralized or decentralized, we assume that a network node that is initially turned on, automatically attempts to establish links with its neighbors. We recall that the main difference between both scenarios is that in centralized solutions, different algorithms could be used for coordinator and follower nodes, if so desired. The basic establishment procedure comprises three main phases:

- **Sensing phase:** The purpose of the sensing phase is to determine the actual state of a given channel prior to accessing it. The main intention behind this is to avoid causing interference to a PU that may be present in the given channel. Depending on the actual scenario and channel dynamics, the sensing phase could be executed only once, at the beginning of the rendezvous procedure, cyclically (i.e., at certain time intervals), or each time the radio tunes to a new channel. It is worth mentioning that during the sensing phase, a node can also receive beacons. It is usually assumed that a single transceiver is used for all phases of the protocol. However, it is also possible to employ a dedicated receiver for spectrum sensing in order to offload the primary transceiver.
- **Beaconing phase:** If no active node has been sensed in its vicinity during the sensing phase or no other rendezvous request beacon has been received, the radio will start transmitting a request beacon on its own in a broadcast fashion in order to reach potential neighbors.
- **Listen phase:** Immediately after transmitting a request beacon, the node enters the listen phase in order to wait for possible response messages. If a node receives a request beacon, it replies to it with a response message within a certain amount of time, indicating that it has received the beacon successfully and wishes to establish a link on the same channel.

[9] All algorithms can be extended to support rendezvous of multiple users as described in [53].

Figure 5.34 shows a basic example with two radios that are hopping over a sequence of channels. They finally meet after six slots in channel 1 and complete the link establishment phase.

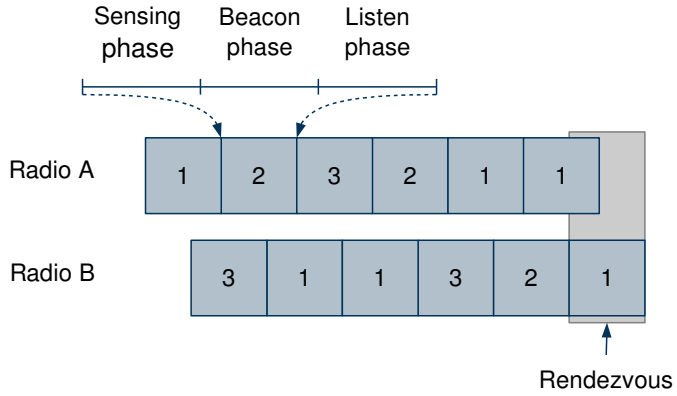


Figure 5.34: *Rendezvous example between two nodes hopping over a sequence of channels, eventually meeting in channel 1 (reproduced from [52]).*

5.5.3.3 System and Component Architecture

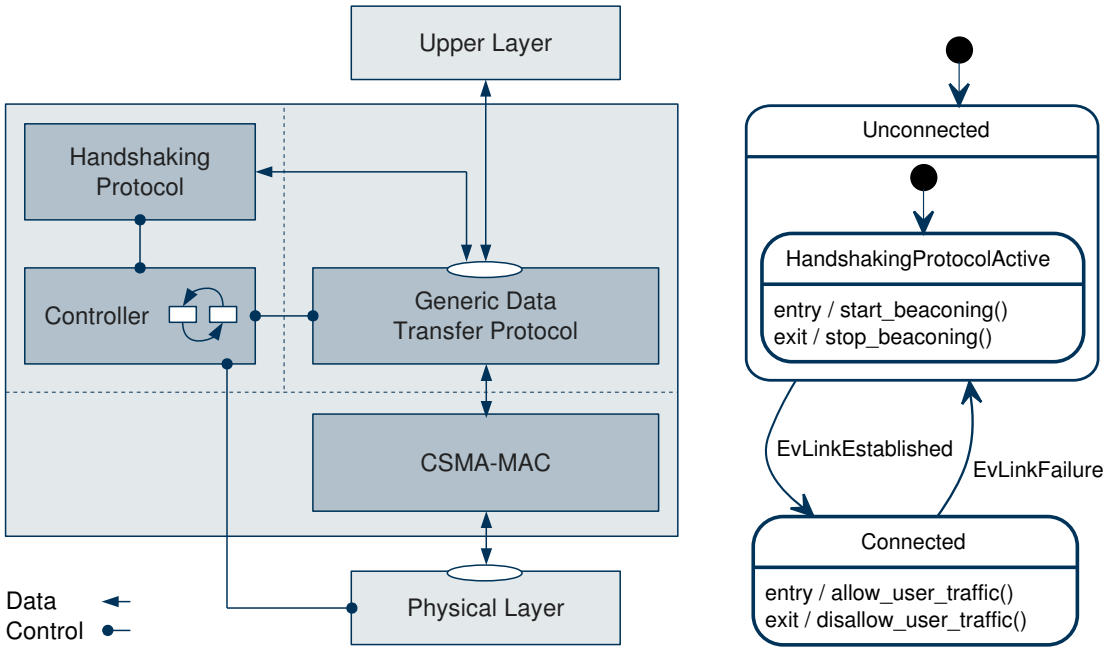
In order to provide link establishment capabilities to an existing link layer protocol, we again need to complement our existing architecture by two more functional blocks:

- **Rendezvous Handshaking Protocol:** This protocol implements the beaconing and handshaking procedure between two nodes that wish to establish a link between one another.
- **Link Establishment Controller:** The controller is responsible for activating and deactivating the handshaking protocol. For this purpose, it is required to understand the difference between a node that has already set up a communication link and one that has not.

Figure 5.35a depicts the high-level design of an *Iris* radio with link establishment capabilities implemented on the basis of the FLL architecture. Precisely speaking, the link establishment controller is rather an extension of an existing FLL controller that extends the operational states of a node. If a node is initially turned on in an opportunistic DSA scenario, it is realistic to assume that it has no knowledge about the current network situation. A number of questions need to be answered in this case, such as: Are there any nodes? If yes, how many? Which channels do they have access to? Thus, it is fair to assume the node to be *unconnected* initially. However, after rendezvous took place and links have been established among nodes, a node may be considered to be *connected* after all.

Within our FLL architecture, this kind of awareness may be realized as an extension of an existing link layer controller that models the operational states of a node. We have realized such a controller for *Iris* that adds this intelligence to a radio by implementing the example HSM that has already been shown in Chapter 4. Both the high-level design and the controller HSM are again shown in Figure 5.35.

However, in addition to that we also require a new link layer application that implements the handshaking procedure, i.e., the beacon exchange protocol, discussed above. We have also



(a) *High-level design with data and control connections.*

(b) *The HSM executed inside the controller.*

Figure 5.35: *High-level design of the radio with link establishment capabilities implemented on the basis of the FLL architecture.*

implemented a component, called *EasyRendezvous*, that is agnostic to the role of the node and the underlying sequence generation algorithm. Therefore, it can be used in centralized as well as in decentralized networks. The actual generation of the CH sequence as well as the radio reconfiguration are implemented inside the controller. The *EasyRendezvous* component and the controller use a simple event-based messaging protocol for communication. If the controller enters the *unconnected* state, it selects the next rendezvous channel and configures the radio accordingly. It then activates the *EasyRendezvous* component by triggering a *EvSendBeacon* event. The protocol instance on the first node then generates and sends a *«RV_SYN»* message. If the controller doesn't receive an *EvLinkEstablished* event within a specific amount of time, it assumes that rendezvous has not occurred and repeats the procedure until a successful rendezvous has happened.

Figure 5.36 shows the message exchange sequence between two nodes that attempt to rendezvous on a common channel. The orange and green colored bars represent the channel that the radios are tuned to. The first *«RV_SYN»* message sent by node 1 is not received by node 2 because both nodes have tuned their radios to different channels. However, the following attempt is successful, indicated through a *«RV_SYN_ACK»* message, because both radios are now tuned to the orange channel.

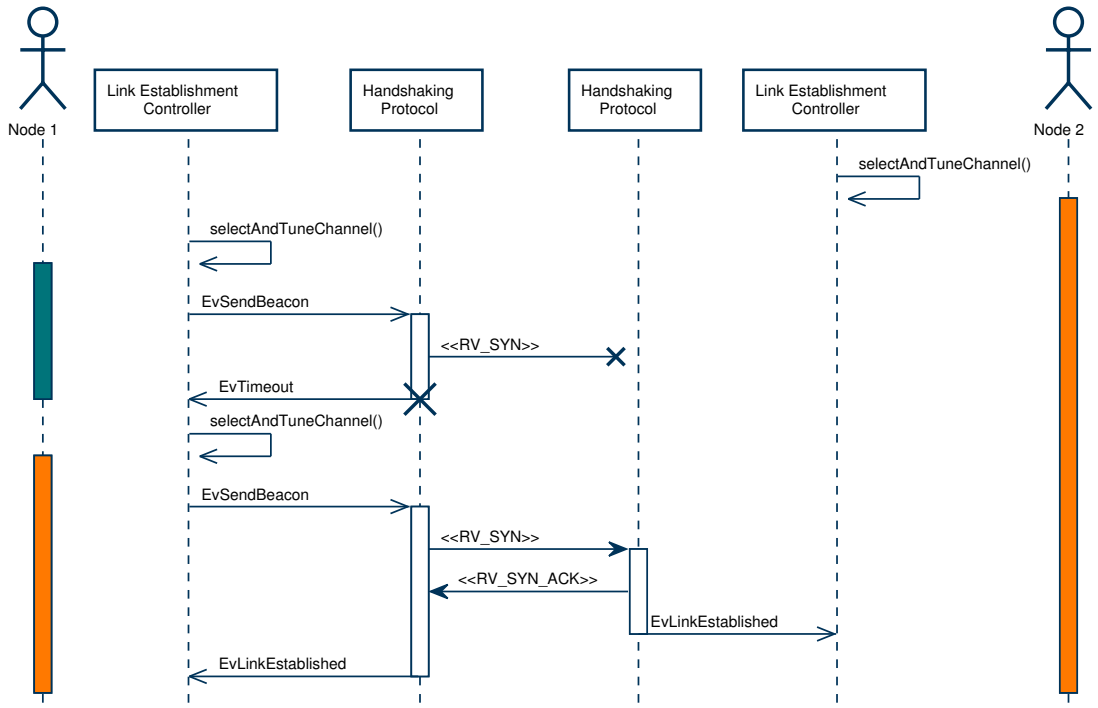


Figure 5.36: *MSC of the rendezvous protocol running between two network nodes wishing to establish a link among each other. The orange and green colored bars represent the channel that the radios are tuned to.*

5.5.3.4 Rendezvous Simulator

Although a great number of rendezvous protocols have been proposed in recent years, there has always been a lack of a common tool for simulating and comparing them with each other. Researchers have either created their own simulators [52] or have used *Matlab* for this purpose [56, 53].

In an attempt to create such a common tool, we have developed an open-source rendezvous algorithm simulator in Python called *RendezvousSim* [169]. On top of this simulator, we have implemented various state-of-the-art algorithms. Specifically, we have implemented random rendezvous [52], sequence-based rendezvous [170], MC and MMC [52], Jump and Stay [53], DRSEQ [171], CRSEQ [61], and EX [56]. To verify the correctness of the implementation, we have reproduced the simulation results published in the original papers. The core features of *RendezvousSim* can be summarized as follows:

- support for centralized and decentralized algorithms,
- support for symmetric and asymmetric channel models, and
- flexible collection and processing of simulation results.

In the following, we employ *RendezvousSim* to evaluate three algorithms for centralized and decentralized networks. Due to the scenario restrictions posed in Section 5.5, we will only con-

Table 5.14: *Experimental time to rendezvous (in slots) using the random algorithm with a varying number of channels (M) after 10 repetitions.*

M	Minimum	Average	Maximum	Standard deviation	95% CI
5	3	5.40	10	2.27	3.87 - 6.92
10	2	6.40	12	3.47	4.06 - 8.73
15	3	9.19	21	3.00	7.16 - 11.19
20	2	10.25	19	5.34	6.66 - 13.84
25	5	13.67	26	6.44	9.33 - 17.98

sider dynamic channels. In other words, we do not include algorithms that rely on a CC. We choose random rendezvous as the baseline algorithm for comparison.

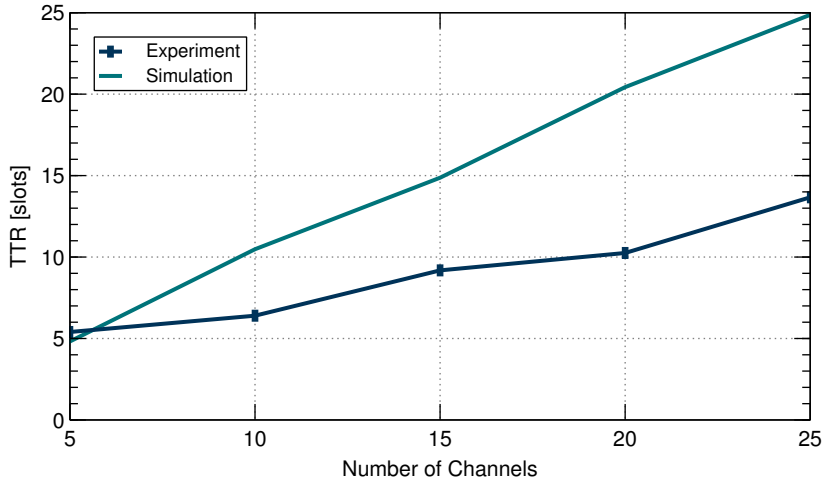
5.5.3.5 Performance Evaluation

We now assess the performance of the link establishment mechanism through both practical experiments and computer simulations. The main purpose of the experimental evaluation is to verify the practical feasibility in general. Computer simulations are then used to compare the performance at a larger scale. Note that we specifically evaluate those two algorithms selected as candidate solutions for centralized and decentralized scenarios in Section 3.2.2, this is exhaustive search (EX) and Jump and Stay (JS).

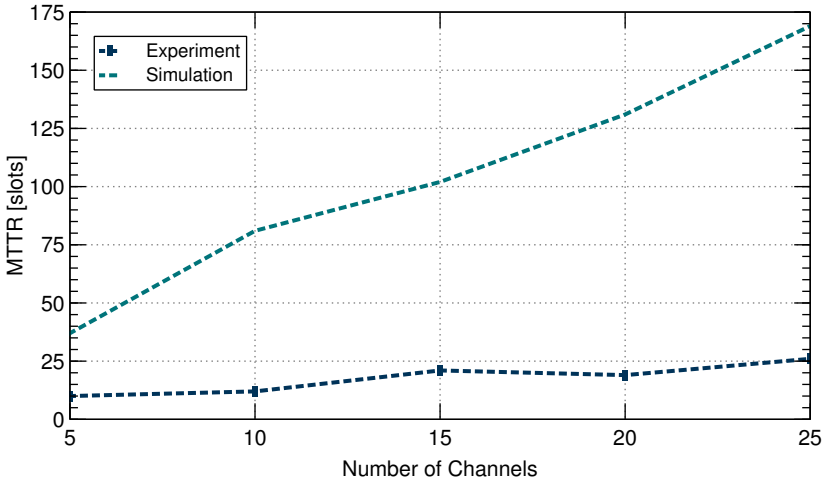
Experimental Results

In order to verify the principle rendezvous functionality within the FLL architecture, we have carried out a number of practical experiments employing the random algorithm. We used the symmetric model and varied the number of potential channels for rendezvous between 5 and 25 with a step size of 5. The beacon interval was set to 500 ms. In each slot, only one rendezvous request beacon was sent. For each value of M, we repeated the experiment 10 times and computed average and maximum TTR. The detailed results are listed in Table 5.14. Figure 5.37 compares experimental and simulation results. Compared to the simulation results, both TTR and MTTR are lower in the experiment, which can be explained by the low number of repetitions. Note that the simulation has been repeated 1.000 times and, thus, achieves the required statistical significance. However, we would like to remind the reader that the sole purpose of this experiment was to verify the principle of operation. We believe the obtained results have successfully demonstrated the link establishment capabilities of the FLL architecture.

We will now extend our study to include more algorithms and scenario parameters. From this point onwards, however, we will only employ computer simulations.



(a) *Average TTR.*



(b) *Maximum TTR.*

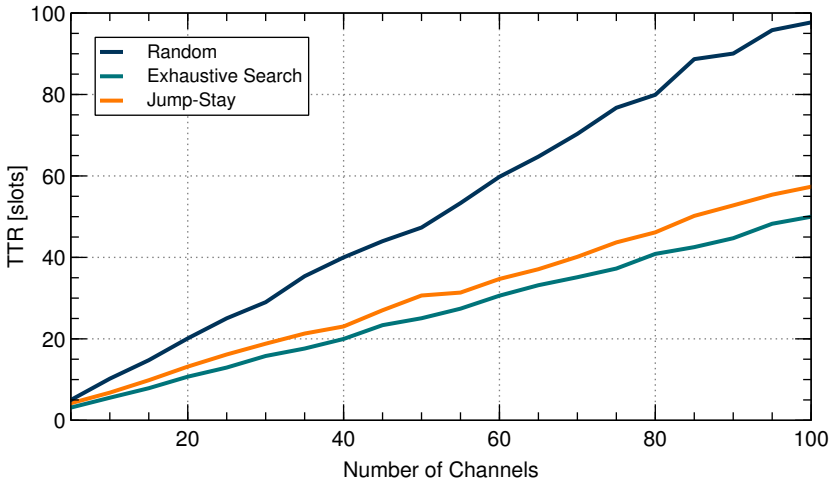
Figure 5.37: *Comparison of experimental and simulated TTR using the random algorithm with increasing number of channels after 10 and 1.000 repetitions, respectively.*

Simulation Results Under The Symmetric Model

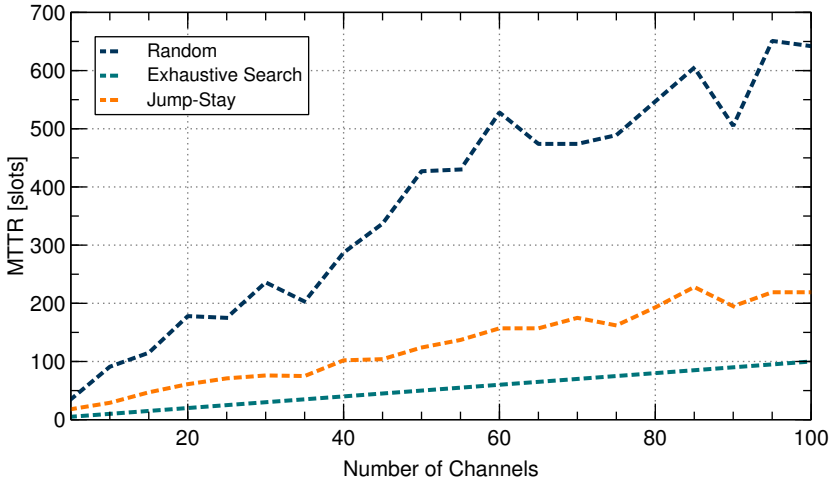
We have simulated both candidate algorithms, EX and JS, selected for centralized and decentralized scenarios, respectively. In each experiment, we have gradually increased the number of channels M available for rendezvous. Specifically, we varied them between 5 and 100 with a step size of 5 and repeated each parameter setting 1.000 times.

Figure 5.38 shows the average and maximum TTR obtained through the simulation. From the results, it can be clearly observed that both TTR and MTTR increase with increasing number of channels, as expected. Overall, EX outperforms both random rendezvous and JS. In the average case, the random algorithm requires approximately M slots and EX $\frac{M}{2}$ slots respectively, for achieving rendezvous. Interestingly, the performance penalty of JS compared to EX decreases

with an increasing number of channels. For example, TTR in a scenario with only 5 channels is 3.12 for EX, and 4.19 for JS, a difference of 34%. However, TTR in a scenario with 100 channels is 49.9 for EX and 57.35 for JS, a difference of merely 14%. A similar relationship can also be observed for maximum TTR.



(a) *Average TTR.*



(b) *Maximum TTR.*

Figure 5.38: *Simulated average and maximum TTR under the symmetric model with increasing number of channels after 1.000 simulation runs.*

Simulation Results Under The Asymmetric Model

We have again simulated both candidate algorithms for centralized and decentralized scenarios. Specifically, we have simulated an environment with $M = 80$ channels and $\theta = 0.5$. The parameter θ has been introduced by the authors of JS as an factor to vary the number of channels

available to each user. Even though the entire set of channels, C, has a cardinality of M, the actual set of channels available to the i-th user, C_i, has a cardinality of only $M \cdot \theta$. In other words, with $\theta = 0.5$, each node has access to $M \cdot \theta = 40$ channels that are randomly chosen from C for each node individually. We vary the number of overlapping channels G between 1 and 20.

Figure 5.39 shows the average and maximum TTR as a function of overlapping channels after 1.000 iterations. Clearly, the more channels both nodes have in common, the faster rendezvous can be achieved. As we can see from the graph, EX again outperforms both other algorithms. As the TTR of EX only depends on the actual number of available channels, but not on the number of common channels, EX only requires 20 slots to achieve rendezvous in the average case (see Figure 5.39a). As has also been reported by the authors of EX, this is exactly half the number of available channels, i.e., $\frac{M \cdot \theta}{2} = \frac{80 \cdot 0.5}{2} = 20$ [56]. In the worst case situation, EX requires exactly M slots to achieve rendezvous, regardless of the number of overlapping channels.

In contrary, the random algorithm and JS benefit from an increasing number of commonly available channels. For example, if the number of overlapping channels is only one, the performance gain of EX may be as high as 2500% compared to JS. However, it should be noted again that EX is only applicable in the decentralized scenario, as it requires explicit roles assigned to the nodes. Interestingly, the random algorithm mostly outperforms JS in both the average and the worst case. But it should also be noted that random CH cannot provide guaranteed rendezvous, as opposed to EX and JS.

5.5.3.6 Summary of Link Establishment

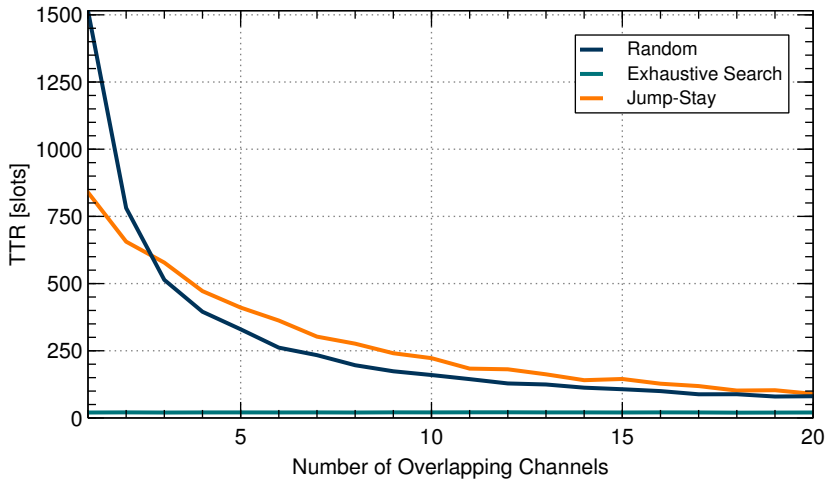
This section has addressed the issue of initially establishing links among nodes in DSA networks. The contribution is twofold: first, we have designed and developed an algorithm-agnostic mechanism for our FLL prototype implementation that has been used to practically verify and demonstrate the link establishment capabilities of the architecture.

Second, motivated by the lack a common simulation framework for rendezvous algorithms, we have designed and implemented our own simulation tool and have integrated various state-of-the-art rendezvous algorithms. Through computer simulations, we have quantified the difference in time needed to establish links using the candidate solutions in centralized networks compared to decentralized networks. We have found that in the symmetric model, the difference between EX and JS is relatively low, i.e., approximately 14% in a radio environment with 100 channels. In contrary, in the asymmetric model, the difference may be quite dramatic, depending on the number of overlapping channels. As EX only depends on the total number of channels, the performance gain may be as high as 2500%, if the number of overlapping channels is very low.

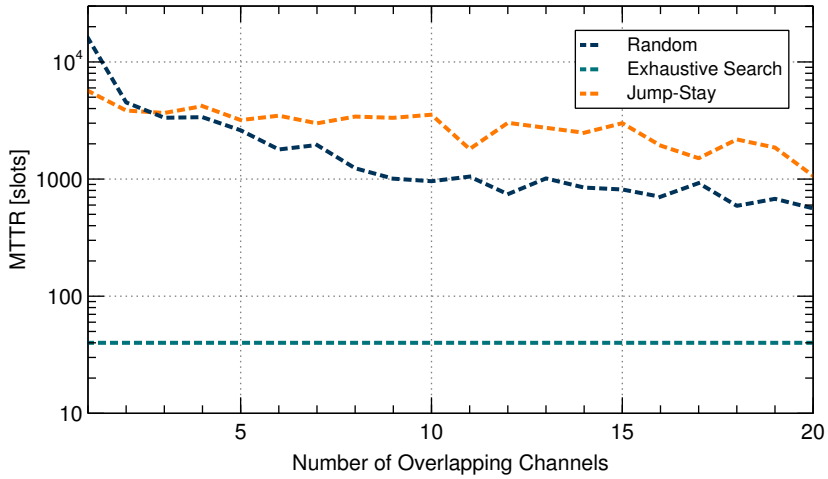
Even though it is difficult to directly apply the obtained results in practice, we believe this analysis still serves as a good basis for further research. For example, the simulation tool could be further extended to include more network topologies and scenarios. It could also be used for studying various kinds of D2D neighbor discovery algorithms for future LTE networks [172].

5.5.4 Summary of Adaptation in Dynamic Spectrum Access Networks

The capability to extend and adapt the system behavior and functionality should be an integral part of FWCS. Based on the envisioned operating environment of a communication system, multiple strategies may be applicable for solving a specific problem or for providing a certain feature.



(a) *Average TTR.*



(b) *Maximum TTR.*

Figure 5.39: *Simulated average and maximum TTR under the asymmetric model ($M = 80$ and $\theta = 0.5$) with an increasing number of overlapping channels after 1.000 simulation runs.*

This section has addressed the issue of providing the necessary degree of link layer flexibility needed to support novel spectrum access paradigms, such as DSA networks. Specifically, we have studied the issue of initially setting up communication links among nodes as well as maintaining them in order to provide seamless communication, even when interrupted by prioritized users. By designing, implementing, and evaluating the required mechanisms both through practical experiments and computer simulations, this section has successfully demonstrated the suitability of the FLL architecture for realizing the envisioned systems.

5.6 Summary

The central objectives of performing practical experiments using real-world prototype implementation of novel protocols or protocol architectures includes the verification of the results obtained through analysis and simulations, and the demonstration of the features claimed by the conceptual model.

We believe that this chapter has successfully proven the suitability of the FLL architecture for building evolving and highly flexible protocols. We began our journey by developing a generic data transfer protocol with basic scheduling, multiplexing, and error control capabilities. Throughout the chapter, this protocol has been extended multiple times by adding additional components, until eventually fulfilling the demands of an opportunistic, decentralized mobile ad-hoc network.

At first, we added a MAC mechanism to the GDTP that allowed multiple radios to share the same communication channel. In fact, we developed a random-access as well as a reservation-based MAC and showed that both have their advantages and disadvantages.

In an attempt to combine the advantages of both MAC strategies in a single system, we then developed a load-adaptive protocol that transparently switches its MAC scheme based on the observed radio performance. The obtained results have shown that the expected performance improvements could be achieved by substituting the media-dependent protocol components only, while keeping the media-independent function in place. Thus, reusability has been successfully demonstrated.

Based on this architecture, we further extended the protocol by adding mechanisms required for DSA networks. Specifically, we added a spectrum mobility protocol that allowed exploiting otherwise unused radio resources in an efficient manner. We showed that the DSA capable link layer configuration was able to provide seamless communication services to its clients, while at the same time protecting the respective owner of the temporarily exploited spectral resource. Lastly, we have illustrated possibilities for realizing link establishment mechanisms on the basis of the FLL architecture.

6 Conclusion

The final chapter of this dissertation summarizes our main findings and contributions and then takes a look on possible areas for future research.

6.1 Summary

Future wireless communication systems (FWCS) are envisioned to support a wide range of application requirements and to function under diverse operating conditions. As a direct consequence, flexibility across the entire life cycle is of paramount importance for the practical realization of evolved and novel communication technologies.

This dissertation sought to explore concepts for developing link layer protocols that accommodate the flexibility for addressing the challenges of FWCS. In the beginning of this thesis, we asked the question whether it is possible to develop a generic architecture for this purpose. Our study has provided an answer to this question by demonstrating the capabilities of the *Flexible Link Layer* as an efficient architecture for building highly adaptive, extensible, and reconfigurable protocols for FWCS.

During the course of the dissertation, we have decomposed existing protocols, analyzed their commonalities and differences, and then suggested a novel concept for recombining them. By means of computer simulations, prototype implementation, and real-world experimentation, we have verified the performance of selected protocols and proved the overall concept viable.

SDR technology has undoubtedly played a major role in this research. Although the practical realization of SDR-based communication systems is by no means a straightforward process, and contains its own difficulties and problems, it served as a useful tool and provided invaluable insight. SDRs are still in their early stages, but can safely be assumed to become more powerful in the near future. However, having to overcome barriers stemming from limitations of current SDR generations also helped to question and rethink fundamental elements and the design process of communication protocols.

The same can also be said of our analysis of actual communication protocols and proposals from the research community. This analysis has identified a number of recurring elements and patterns that are often combined in a monolithic fashion. We believe that those fundamental elements should not be replicated multiple times within a single layer. Furthermore, we think

related functions of a layer should be implemented close to each other. In contrary, we suggest designs that keep unrelated functions of a protocol as far separated from each other as possible. By following this "separation of concerns" philosophy, the link layer architecture developed in this dissertation allowed us to realize numerous practically working protocols, implement and verify algorithms and ideas, and, ultimately, to experience and experiment with technology that will be part of our future life. We strongly believe that this wouldn't have been possible without the contributions of this thesis.

However, our study also revealed limitations that are still present and need to be investigated in the future. Admittedly, the architecture proposed in this dissertation and the resulting discussions are oriented towards DSA networks, which represent only one technology component of FWCS. The study, therefore, needs to be expanded in order to prove its applicability also to other technology components and application domains, such as next-generation cellular networks or M2M-type communication networks. Nonetheless, we believe it already provides a useful tool, especially for experimental-driven research, that may assist us in shaping the future of wireless communication.

6.2 Future Work

In developing the FLL architecture, this dissertation has also discovered a number of open research questions and possible areas for further investigations. This section discusses some of these issues.

- **Application to other Domains:** As has already been mentioned above, the application of the FLL architecture and its underlying concepts to other communication technologies, such as cellular networks, IoT, or vehicular communication, would be an exciting research field. Very recently, device-to-device communication in infrastructure-based networks has seen an enormous amount of interest in both academia and industry. We believe that protocol suites for both user equipment and base stations could benefit from a flexible protocol architecture. Furthermore, it would be very interesting to extend the study to underlay DSA networks.

- **Integration of the Recursive Inter-Network Architecture:** The recursive inter-network architecture (RINA) is a novel network architecture based on the assumption that a network layer is a recurring element with similar functions but different scope. The concept has been developed by John Day [38] and has been a subject of research for some time due to its novel perception of what networking is. Even though the FLL architecture builds upon and exploits some of the concepts that can also be found in RINA, we didn't explicitly design our architecture as "RINA for wireless networks". However, there are some indications that the FLL may be extended towards this direction, but further research is certainly required.

- **Design of a Generic Feature Negotiation Protocol:** The FLL architecture employs mechanisms for negotiating certain features or parameters between protocol instances at various places. Those mechanisms are currently designed for specific parameters. However, it can be seen that quite some similarities exist between them. Thus, future work could study the

issue of designing a generic policy selection, or feature negotiation, protocol that could be shared among different parameters.

- **Link Layer Protocols for Higher Efficiency and Reliability:** During our analysis, we have discussed a number of shortcomings of IEEE 802.11-based protocols that increase their complexity, limit their efficiency, and make them non-deterministic. However, communication requirements, such as predictability and fault-tolerance, will be key for many future applications. For example, some safety-critical use cases for vehicular networks, such as autonomous driving will certainly require more reliable communication systems. Therefore, future research could investigate link layer protocols which are suitable for such applications, possibly also incorporating DSA access strategies for redundant communication.

List of Abbreviations

ACK	Acknowledgment
ADC	Analog-to-Digital Converter
AGC	Automatic Gain Control
AM	Acknowledged Mode
AP	Access Point
API	Application Programming Interface
ARQ	Automatic Repeat Request
ASIC	Application-Specific Integrated Circuit
BA	Block Acknowledgement
BC	Backup Channel
BDP	Bandwidth-Delay Product
BP	Beacon Period
CBF	Channel Busy Fraction
CBR	Constant Bit Rate
CC	Control Channel
CCA	Clear Channel Assessment
CDMA	Code-Division Multiple Access
CDT	Channel Detection Time
CH	Channel Hopping
CI	Confidence Interval
CMP	Coordinator Mobility Protocol
CP	Composite Protocols
CPC	Cognitive Pilot Channel
CR	Cognitive Radio
CRSEQ	Channel Rendezvous Sequence
CSMA	Carrier Sense Multiple Access
CTP	Configurable Transport Protocol
CTS	Clear To Send

CTSMC Continuous-Time Semi Markov Chain
CW Contention Window

D2D Device-to-Device
DAC Digital-to-Analog Converter
DC Duty Cycle
DCF Distributed Coordination Function
DECT Digital Enhanced Cordless Telecommunications
DIFS DCF Interframe Space
DSA Dynamic Spectrum Access
DSP Digital Signal Processor
DTP Data Transfer Period

ED Energy Detector
EFCP Error and Flow Control Protocol
EX Exhaustive Search

FCC Federal Communications Commission
FDMA Frequency-Division Multiple Access
FEC Forward Error Correction
FER Frame Error Rate
FIFO First In First Out
FLL Flexible Link Layer
FPGA Field-Programmable Gate Array
FSM Finite State Machine
FWCS Future Wireless Communication Systems

GDTP Generic Data Transfer Protocol
GLL Generic Link Layer
GPD Generalized Pareto Distribution
GPL General Public License
GPP General-Purpose Processor
GR GNU Radio
GSM Global System for Mobile Communications

HC High Contention
HDLC High Level Data Link Control
HNP Handover Negotiation Protocol
HSM Hierarchical State Machine
HW Hardware

I/Q In-phase/Quadrature
ID Identifier
IEEE Institute of Electrical and Electronics Engineers
IF Intermediate Frequency
IoT Internet of Things
IP Internet Protocol
ISM Industrial, Scientific and Medical

JS Jump and Stay

LAN Local Area Network
LC Low Contention
LLA Link Layer Application
LLC Logical Link Control
LNA Low-Noise Amplifier
LTE Long Term Evolution

M2M Machine-to-Machine
MAC Medium Access Control
MC Markov Chain or Modified Clock
MIMO Multiple-Input and Multiple-Output
MMC Modified Modular Clock
MP Micro Protocol
MSC Message Sequence Chart
MSL Maximum Segment Lifetime
MTTR Maximum Time To Rendezvous
MTU Maximum Transfer Unit

NIC Network Interface Card

OFDM Orthogonal Frequency-Division Multiplexing
OSI Open Systems Interconnection
OTA Over The Air

PA Power Amplifier
PC Personal Computer
PCI Protocol Control Information
PDU Protocol Data Unit
PHY Physical

PoA Point of Attachment
PPS Pulse Per Second
PU Primary User

QoS Quality of Service

RAT Radio Access Technology
RC Rendezvous Channel
RF Radio Frequency
RINA Recursive Inter-Network Architecture
RR Round Robin
RTS Request To Send

SDMA Space-Division Multiple Access
SDR Software Defined Radio
SDU Service Data Unit
SF Super Frame
SIFS Short Interframe Space
SNR Signal-to-Noise Ratio
SoC Separation of Concerns
STA Station
STL Standard Template Library
SU Secondary User
SW Software

TC Traffic Categories
TCP Transmission Control Protocol
TDMA Time-Division Multiple Access
TETRA Terrestrial Trunked Radio
TS Time Synchronization
TTR Time To Rendezvous

UDP User Datagram Protocol
UHD USRP Hardware Driver
UM Unacknowledged Mode
UML Unified Modeling Language
UMTS Universal Mobile Telecommunications System
USRP Universal Software Radio Peripheral

WARP Wireless Open-Access Research Platform

WE Wiring Engine
WLAN Wireless Local Area Network
WOSA Weighted Overlapped Segments Averaging
WSN Wireless Sensor Network

XML Extensible Markup Language

ZDKS Zukünftige drahtlose Kommunikationssysteme

List of Figures

List of Tables

Bibliography

[1] Andrea Goldsmith. *Wireless Communications*. Cambridge University Press, 2005.

[2] Cisco. Cisco Visual Networking Index: Forecast and Methodology, 2013-2018, June 2014.
Available under `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.pdf`; last visited on August 30, 2014.

[3] Robert Baldemair, Erik Dahlman, Gabor Fodor, Gunnar Mildh, Stefan Parkvall, Yngve Selén, Hugo Tullberg, and Kumar Balachandran. Evolving Wireless Communications: Addressing the Challenges and Expectations of the Future. *IEEE Vehicular Technology Magazine*, 8(1):24–30, March 2013.

[4] André Puschmann and Andreas Mitschele-Thiel. Modeling Management Functions as Link Layer Applications. In *Networked Systems (NetSys)*, pages 1–2, Cottbus, Germany, March 2015.

[5] André Puschmann and Andreas Mitschele-Thiel. Implementation and Evaluation of a Flexible, Load-Adaptive Link Layer Protocol. In *20th Annual International Conference on Mobile Computing and Networking (MobiCom), 9th ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH)*, pages 73–80, Maui, HI, USA, September 2014.

[6] André Puschmann, Paolo Di Francesco, Mohamed A. Kalil, Luiz A. DaSilva, and Andreas Mitschele-Thiel. Enhancing the Performance of Random Access MAC Protocols for Low-cost SDRs. In *19th Annual International Conference on Mobile Computing and Networking (MobiCom), 8th International Workshop on Wireless Network Testbeds Experimental Evaluation and Characterization (WiNTECH)*, pages 9–16, Miami, FL, USA, September 2013.

[7] André Puschmann, Mohamed A. Kalil, and Andreas Mitschele-Thiel. A Component-based Approach for Constructing Flexible Link-Layer Protocols. In *8th International Conference on Cognitive Radio Oriented Wireless Networks (CROWNCOM)*, pages 244–249, Washington D.C., USA, July 2013.

[8] André Puschmann, Shah N. Khan, Mohamed A. Kalil, and Andreas Mitschele-Thiel. Database-assisted Coordinator-based Spectrum Mobility in Cognitive Radio Ad-hoc Net-

works. In *10th International Symposium on Wireless Communication Systems (ISWCS), Ilmenau, Germany, August 2013*, pages 1–2, Ilmenau, Germany, August 2013.

[9] André Puschmann, Mohamed A. Kalil, and Andreas Mitschele-Thiel. A Flexible CSMA based MAC Protocol for Software Defined Radios. *Frequenz Journal of RF-Engineering and Telecommunications*, 6:261–268, October 2012.

[10] André Puschmann, Mohamed A. Kalil, and Andreas Mitschele-Thiel. Implementation and Evaluation of a Practical SDR Testbed. In *4th International Conference on Cognitive Radio and Advanced Spectrum Management (COGART)*, pages 1–5, Barcelona, Spain, October 2011.

[11] Bastian Bloessl, André Puschmann, Christoph Sommer, and Falko Dressler. Timings Matter: Standard Compliant IEEE 802.11 Channel Access for a Fully Software-based SDR Architecture. In *20th Annual International Conference on Mobile Computing and Networking (MobiCom), 9th ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH)*, pages 57–64, Maui, HI, USA, September 2014.

[12] Tobias Simon, André Puschmann, Shah N. Khan, and Andreas Mitschele-Thiel. A Lightweight Message-based Inter-Component Communication Infrastructure. In *5th International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN2013)*, pages 145–152, Madrid, Spain, June 2013.

[13] André Puschmann and Mohamed A. Kalil. The Impact of a Dedicated Sensing Engine on a SDR Implementation of the CSMA Protocol. In *10th International Symposium on Wireless Communication Systems (ISWCS)*, pages 1–5, Ilmenau, Germany, August 2013.

[14] André Puschmann, Shah N. Khan, Ali H. Mahdi, Mohamed A. Kalil, and Andreas Mitschele-Thiel. An Architecture for Cognitive Radio Ad-Hoc Network Nodes. In *12th International Symposium on Communications and Information Technologies (ISCIT)*, pages 1–5, Gold Coast, Australia, October 2012.

[15] Mohamed A. Kalil, André Puschmann, and Andreas Mitschele-Thiel. SWITCH: A Multichannel MAC Protocol for Cognitive Radio Ad Hoc Network. In *IEEE 76th Vehicular Technology Conference (VTC)*, pages 1–5, Québec City, Canada, September 2012.

[16] Geoffrey Ye Li, Zhikun Xu, Cong Xiong, Chenyang Yang, Shunqing Zhang, Yan Chen, and Shugong Xu. Energy-Efficient Wireless Communications: Tutorial, Survey, and Open Issues. *IEEE Wireless Communications*, 18(6):28–35, December 2011.

[17] Douglas C. Schmidt, Donald F. Box, and Tatsuya Suda. ADAPTIVE: A Dynamically Assembled Protocol Transformation, Integration, and eValuation Environment. *Journal of Concurrency: Practice and Experience*, 5(4):269–286, June 1993.

[18] Matthias Wellens, Jin Wu, and Petri Mähönen. Evaluation of Spectrum Occupancy in Indoor and Outdoor Scenario in the Context of Cognitive Radio. In *2nd International Conference on Cognitive Radio Oriented Wireless Networks and Communications (CROWNCOM)*, pages 420–427, Orlando, FL, USA, August 2007.

[19] Ian F. Akyildiz, Won-Yeol Lee, Mehmet C. Vuran, and Shantidev Mohanty. NeXt Generation/Dynamic Spectrum Access/Cognitive Radio Wireless Networks: A Survey. *Compututer Network*, 50(13):2127–2159, September 2006.

[20] Joseph Mitola III. *Cognitive Radio An Integrated Agent Architecture for Software Defined Radio*. PhD thesis, Royal Institute of Technology (KTH), 2000.

[21] SDR Forum Cognitive Radio Working Group and others. Cognitive Radio Definitions and Nomenclature (SDRF-06-P-0009-V1.0.0), 2008.

[22] Simon Haykin. Cognitive Radio: Brain-empowered Wireless Communications. *IEEE Journal on Selected Areas in Communications*, 23(2), February 2005.

[23] Alexander M. Wyglinski, Maziar Nekovee, and Y. Thomas Hou, editors. *Cognitive Radio Communications and Networks: Principles and Practice*. Elsevier Academic Press, 1. edition, November 2009.

[24] Coleman O'Sullivan. *Deconstructing and Reconstructing Medium Access Control: The Anatomy of a Cognitive Radio MAC*. PhD thesis, Tinity College Dublin, Ireland, 2012.

[25] Linda E. Doyle. *Essentials of Cognitive Radio*. Cambridge University Press, 2009.

[26] Ezio Biglieri, Andrea J. Goldsmith, Larry J. Greenstein, Narayan B. Mandayam, and H. Vincent Poor. *Principles of Cognitive Radio*. Cambridge University Press, 2012.

[27] 3rd Generation Partnership Project (3GPP). LTE in unlicensed spectrum.
Available under `http://www.3gpp.org/news-events/3gpp-news/1603-lte_in_unlicensed`; last visited on October 16, 2014.

[28] Tony J. Rouphael. *RF and Digital Signal Processing for Software-Defined Radio: A System-Analytic Approach*. A Multi-Standard Multi-Mode Approach Series. Newnes, Elsevier Science, 2009.

[29] Joseph Mitola III. Software Radios: Survey, Critical Evaluation and Future Directions. In *National Telesystems Conference (NTC)*, pages 13/15–13/23, 1992.

[30] Bruce A. Fette, editor. *Cognitive Radio Technology*. Elsevier Science, 2nd edition, 2009.

[31] Ettus Research. Website. Available under `http://www.ettus.com`; last visited on January 14, 2014.

[32] Rahul Dhar, Gesly George, Amit Malani, and Peter Steenkiste. Supporting Integrated MAC Protocol and PHY Software Development for the USRP SDR. In *1st IEEE Workshop on Networking Technologies for Software Defined Radio Networks*, Reston, VA, USA, September 2006.

[33] Stefan Valentin, Holger von Malm, and Holger Karl. Evaluating the GNU Software Radio Platform For Wireless Testbeds. Technical report, University of Paderborn, Department of Computer Science, 2006.
Available under `http://nesl.ee.ucla.edu/fw/thomas/wintech07.pdf`; last visited on May 5, 2014.

[34] Thomas Schmid, Oussama Sekkat, and Mani B. Srivastava. An Experimental Study of Network Performance Impact of Increased Latency in SDR. In *ACM Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WINTECH)*, Montréal, Québec, Canada, September 2007.

[35] George Nychis, Thibaud Hottelier, Zhuochen Yang, Srinivasan Seshan, and Peter Steenkiste. Enabling MAC Protocol Implementations on Software-defined Radios. In *6th USENIX Symposium on Networked Systems Design and Implementation*, Boston, MA, USA, April 2009.

[36] Andrew Tanenbaum. *Computer Networks*. Prentice Hall Professional Technical Reference, 4th edition, 2002.

[37] Institute of Electrical and Electronics Engineers. *IEEE 802.11ac Standard: Amendment 4: Enhancements for Very High Throughput for Operation in Bands below 6 GHz*, December 2013.

[38] John Day. *Patterns in Network Architecture: A Return to Fundamentals*. Prentice Hall, 2008.

[39] Guo-Yuan Mikko Wang and Chunhung Richard Lin. Link layer Assisted IP Mobility Protocol. In *13th IEEE International Conference on Networks*, Kuala Lumpur, Malaysia, November 2005.

[40] Charles E. Perkins on behalf of the Internet Engineering Task Force (IETF). RFC5944: IP Mobility Support for IPv4, November 2010.

[41] David A. Maltz and Pravin Bhagwat. MSOCKS: An Architecture for Transport Layer Mobility. In *17th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 3, pages 1037–1045, San Francisco, CA, USA, 1998.

[42] Vatche Ishakian, Joseph Akinwumi, Flavio Esposito, and Ibrahim Matta. On Supporting Mobility and Multihoming in Recursive Internet Architectures. Technical report, Computer Science Department, Boston University, 2010.

[43] Eleni Trouva, Eduard Grasa, John Day, Ibrahim Matta, Lou Chitkushev, Steve Bunch, Miguel Ponce de Leon, Patrick Phelan, and Xavier Hesselbach-Serra. Transport over Heterogeneous Networks Using the RINA Architecture. In *9th International Conference on Wired/Wireless Internet Communications (WWIC)*, Vilanova i la Geltrú, Catalonia, Spain, June 2011.

[44] En-Yi A. Lin, Jan M. Rabaey, and Adam Wolisz. Power-efficient Rendez-vous Schemes for Dense Wireless Sensor Networks. In *IEEE International Conference on Communications (ICC)*, volume 7, pages 3769–3776 Vol.7, Paris, France, 2004.

[45] Giuseppe Anastasi, Marco Conti, Mario Di Francesco, and Andrea Passarella. Energy Conservation in Wireless Sensor Networks: A Survey. *Ad Hoc Networks*, 7(3):537–568, 2009.

[46] Hoi-Sheung Wilson So, Jean Walrand, and Jeonghoon Mo. McMAC: A Parallel Rendezvous Multi-Channel MAC Protocol. In *IEEE Wireless Communications and Networking Conference (WCNC)*, pages 334–339, Hong Kong, March 2007.

[47] Brandon F. Lo. A Survey of Common Control Channel Design in Cognitive Radio Networks. *Physical Communication*, 4(1):26–39, March 2011.

[48] Rolf Neugebauer and Derek McAuley. Energy is Just Another Resource: Energy Accounting and Energy Pricing in the Nemesis OS. In *8th Workshop on Hot Topics in Operating Systems*, pages 67–72, Elmau, Germany, May 2001.

[49] Niky Riga., Ibrahim Matta, Alberto Medina, Craig Partridge, and Jason Redi. An Energy-conscious Transport Protocol for Multi-hop Wireless Networks. In *ACM Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, pages 1–12, New York, NY, USA, December 2007.

[50] Niky Riga. *JTP, An Energy-Aware Transport Protocol For Mobile Ad Hoc Networks*. PhD thesis, Boston University, 2013.

[51] Institute of Electrical and Electronics Engineers. *IEEE 802.11 Standard, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, June 2007.

[52] Nick C. Theis, Ryan W. Thomas, and Luiz A. DaSilva. Rendezvous for Cognitive Radios. *IEEE Transactions on Mobile Computing*, 10(2), February 2011.

[53] Hai Liu, Zhiyong Lin, Xiaowen Chu, and Yiu-Wing Leung. Jump-Stay Rendezvous Algorithm for Cognitive Radio Networks. *IEEE Transactions on Parallell and Distributed Systems*, 23(10):1867–1881, 2012.

[54] Javier Gozalvez. Long-Term Evolution Direct: A Device-to-Device Discovery Platform. *IEEE Vehicular Technology Magazine*, 9(3):10–17, September 2014.

[55] Hai Liu, Zhiyong Lin, Xiaowen Chu, and Y.-W. Leung. Taxonomy and Challenges of Rendezvous Algorithms in Cognitive Radio Networks. In *International Conference on Computing, Networking and Communications (ICNC)*, pages 645–649, 2012.

[56] Yogesh R. Kondareddy, Prathima Agrawal, and Krishna Sivalingam. Cognitive Radio Network setup without a Common Control Channel. In *IEEE Military Communications Conference (MILCOM)*, pages 1–6, San Diego, CA, USA, November 2008.

[57] Haile B. Weldu, Bosung Kim, and Byeong-hee Roh. Deterministic Approach to Rendezvous Channel Setup in Cognitive Radio Networks. In *International Conference on Information Networking (ICOIN)*, pages 690–695, Bangkok, Thailand, January 2013.

[58] Jordi Pérez-Romero, Oriol Sallent, Ramón Agustí, and Lorenza Giupponi. A Novel On-Demand Cognitive Pilot Channel Enabling Dynamic Spectrum Allocation. In *2nd IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN)*, pages 46–54, Dublin, Ireland, April 2007.

[59] Paul D. Sutton, Keith E. Nolan, and Linda E. Doyle. Cyclostationary Signatures in Practical Cognitive Radio Applications. *IEEE Journal on Selected Areas in Communications*, 26(1):13–24, January 2008.

[60] Paul D. Sutton. *Rendezvous And Coordination in OFDM-based Dynamic Spectrum Access Networks*. PhD thesis, Tinity College Dublin, Ireland, 2008.

[61] Jongmin Shin, Dongmin Yang, and Cheeha Kim. A Channel Rendezvous Scheme for Cognitive Radio Networks. *IEEE Communications Letters*, 14(10):954–956, 2010.

[62] Zhiyong Lin, Hai Liu, Xiaowen Chu, and Yiu-Wing Leung. Enhanced Jump-Stay Rendezvous Algorithm for Cognitive Radio Networks. *IEEE Communications Letters*, 17(9):1742–1745, September 2013.

[63] Ivan Christian, Sangman Moh, Ilyong Chung, and Jinyi Lee. Spectrum Mobility in Cognitive Radio Networks. *IEEE Communications Magazine*, 50(6):114–121, June 2012.

[64] Yonghong Zeng, Ying-Chang Liang, AnhTuan Hoang, and Rui Zhang. A Review on Spectrum Sensing for Cognitive Radio: Challenges and Solutions. *EURASIP Journal on Advances in Signal Processing*, 2010(1):1–16, January 2010.

[65] A Ghasemi and E.S. Sousa. Spectrum Sensing in Cognitive Radio Networks: Requirements, Challenges and Design Trade-offs. *IEEE Communications Magazine*, 46(4):32–39, April 2008.

[66] Jihoon Park, Przemyslaw Pawelczak, , and Danijela Cabric. To Buffer or to Switch: Design of Multichannel MAC for OSA Ad Hoc Networks. In *IEEE Symposium on New Frontiers in Dynamic Spectrum (DySPAN)*, Singapore, April 2010.

[67] Yan Zhang. Spectrum Handoff in Cognitive Radio Networks: Opportunistic and Negotiated Situations. In *IEEE International Conference on Communications (ICC)*, pages 1–6, Dresden, Germany, June 2009.

[68] Maurice Waite, editor. *Oxford Dictionary and Thesaurus*. Oxford University Press, 2nd edition, 2007.

[69] Ian Sommerville. *Software Engineering*. Addison Wesley, 9th edition, 2010.

[70] Giuseppe Bianchi, Pierluigi Gallo, Domenico Garlisi, Fabrizio Giuliano, Francesco Gringoli, and Ilenia Tinnirello. MAClets: Active MAC Protocols over Hard-Coded Devices. In *8th International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, pages 229–240, Nice, France, December 2012.

[71] Ashish Sharma and Elizabeth Belding. FreeMAC: Framework for Multi-Channel MAC Development on 802.11 Hardware. In *ACM SIGCOMM Workshop on Programmable Routers for Extensible Services of Tomorrow (PRESTO)*, pages 69–74, Seattle, WA, USA, August 2008.

[72] Christian Doerr, Michael Neufeld, Jeff Fifield, Troy Weingart, Douglas C. Sicker, and Dirk Grunwald. MultiMAC - An Adaptive MAC Framework for Dynamic Radio Networking. In *1st IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN)*, pages 548–555, Baltimore, MD, USA, November 2005.

[73] Ahmed Khattab, Joseph Camp, Chris Hunter, Patrick Murphy, Ashutosh Sabharwal, and Edward W. Knightly. WARP: A Flexible Platform for Clean-slate Wireless Medium Access Protocol Design. *ACM SIGMOBILE Mobile Computing and Communications Review*, 12(1):56–58, January 2008.

[74] Wireless Open-Access Research Platform developers. WARP project. Available under `http://warpproject.org/trac/wiki/about`.

[75] Wireless Open-Access Research Platform developers. 802.11 Reference Design Architecture. Available under `http://warpproject.org/trac/wiki/802.11/Architecture`; last visited on May 5, 2014.

[76] Paolo Di Francesco, Séamas McGettrick, Uchenna K. Anyanwu, J. Colman O'Sullivan, Allen B. MacKenzie, and Luiz A. DaSilva. A Split MAC Approach for SDR Platforms. *IEEE Transactions on Computers*, PP(99):1–14, February 2014.

[77] Felipe Cerqueira and Björn B. Brandenburg. A Comparison of Scheduling Latency in Linux, PREEMPT-RT, and LITMUS-RT. In *9th Annual Workshop on Operating Systems Platforms for Embedded Real-Time applications (OSPERT)*, pages 19–29, Paris, France, July 2013.

[78] Kun Tan, He Liu, Jiansong Zhang, Yongguang Zhang, Ji Fang, and Geoffrey M. Voelker. Sora: High-performance Software Radio Using General-purpose Multi-core Processors. *Communications of the ACM*, 54(1):99–107, January 2011.

[79] Microsoft Research Software Radio developers. SORA project. Available under `http://research.microsoft.com/en-us/projects/sora/`; last visited on May 5, 2014.

[80] Paolo Di Francesco, Séamas McGettrick, Uchenna K. Anyanwu, J. Colman O'Sullivan, Allen B. MacKenzie, and Luiz A. DaSilva. A Split Architecture for Random Access MAC for SDR Platforms. In *8th International Conference on Cognitive Radio Oriented Wireless Networks (CROWNCOM)*, pages 250–255, Washington D.C., USA, July 2013.

[81] Bastian Bloessl, Michele Segata, Christoph Sommer, and Falko Dressler. Towards an Open Source IEEE 802.11p Stack: A Full SDR-based Transceiver in GNURadio. In *5th IEEE Vehicular Networking Conference (VNC)*, pages 143–149, Boston, MA, USA, December 2013.

[82] Paul D. Sutton, Jörg Lotze, Hicham Lahlou, Suhaib A. Fahmy, Keith E. Nolan, Baris Özgül, Thomas W. Rondeau, Juanjo Noguera, and Linda E. Doyle. Iris: An Architecture for Cognitive Radio Networking Testbeds. *IEEE Communications Magazine*, 48(9):114–122, September 2010.

[83] Software Radio Systems Ltd. The Iris project page. Available under `http://www.softwareradiosystems.com/redmine/projects/iris`; last visited on April 15, 2014.

[84] David G. Messerschmitt. Rethinking Components: From Hardware and Software to Systems. *Proceedings of the IEEE*, 95(7):1473–1496, July 2007.

[85] James M. Bieman and Byung-Kyoo Kang. Measuring Design-Level Cohesion. *IEEE Transactions on Software Engineering*, 24(2):111–124, February 1998.

[86] Christian Kästner. *Virtual Separation of Concerns: Toward Preprocessors 2.0*. PhD thesis, Otto-von-Guericke-Universität Magdeburg, Magdeburg, Germany, May 2010.

[87] Edsger W. Dijkstra. On The Role of Scientific Thought. 1974.

[88] Derek Greer. The Art of Separation of Concerns, January 2003. Available under `http://aspiringcraftsman.com/2008/01/03/art-of-separation-of-concerns/`; last visited on October 22, 2014.

[89] Patrick G. Bridges, Gary T. Wong, Matti Hiltunen, Richard D. Schlichting, and Matthew J. Barrick. A Configurable and Extensible Transport Protocol. *IEEE/ACM Transactions on Networking*, 15(6):1254–1265, December 2007.

[90] Hari Balakrishnan, Srinivasan Seshan, Elan Amir, and Randy H. Katz. Improving TCP/IP Performance over Wireless Networks. In *1st Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 2–11, New York, NY, USA, November 1995.

[91] Nicolas Van Wambeke, Ernesto Exposito, Christophe Chassot, and Michel Diaz. ATP: A Microprotocol Approach to Autonomic Communication. *IEEE Transactions on Computers*, 62(11):2131–2140, November 2013.

[92] Robert Braden, Ted Faber, and Mark Handley. From Protocol Stack to Protocol Heap: Role-based Architecture. *SIGCOMM Computer Communication Review*, 33(1):17–22, January 2003.

[93] Norman C. Hutchinson and Larry L. Peterson. The x-Kernel: An Architecture for Implementing Network Protocols. *IEEE Transactions on Software Engineering*, 17(1):64–76, January 1991.

[94] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. The Click Modular Router. *ACM Transactions on Computer Systems (TOCS)*, 18(3):263–297, August 2000.

[95] Magesh Kannan, Ed Komp, Gary Minden, and Joseph Evans. Design and Implementation of Composite Protocols. Technical Report ITTC-FY2003-TR-19740-05, Information Telecommunication and Technology Center, University of Kansas, Lawrence, KS, USA, February 2003.

[96] John Day, Ibrahim Matta, and Karim Mattar. Networking is IPC: A Guiding Principle to a Better Internet. In *Re-Architecting the Internet (ReArch)*, Madrid, Spain, December 2008.

[97] Kevin Klues, Gregory Hackmann, Octav Chipara, and Chenyang Lu. A Component-Based Architecture for Power-Efficient Media Access Control in Wireless Sensor Networks. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Sydney, Australia, November 2007.

[98] Joseph Polastre, Jonathan Hui, Philip Levis, Jerry Zhao, David Culler, Scott Shenker, and Ion Stoica. A Unifying Link Abstraction for Wireless Sensor Networks. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, San Diego, CA, USA, November 2005.

[99] Joachim Sachs. A Generic Link Layer for Future Generation Wireless Networking. In *IEEE International Conference on Communications (ICC)*, Anchorage, AK, USA, May 2003.

[100] Lars Berlemann. *Distributed Quality-of-Service Support in Cognitive Radio Networks*. PhD thesis, RWTH Aachen, Aachen, Germany, 2006.

[101] K. Mandke, Soon-Hyeok Choi, Gibeom Kim, R. Grant, R.C. Daniels, Wonsoo Kim, R.W. Heath, S.M. Ketan Mandke Nettles, Soon-Hyeok Choi, Gibeom Kim, Robert Grant, Robert C. Daniels, Wonsoo Kim, Jr. Robert W. Heath, and Scott M. Nettles. Early Results on Hydra: A Flexible MAC/PHY Multihop Testbed. In *IEEE 65th Vehicular Technology Conference (VTC)*, pages 1896–1900, Dublin, Ireland, April 2007.

[102] Lei Yang, Zengbin Zhang, Wei Hou, Ben Y. Zhao, and Haitao Zheng. Papyrus: A Software Platform for Distributed Dynamic Spectrum Sharing Using SDRs. *ACM Computer Communication Review (CCR)*, 41(1), January 2011.

[103] Junaid Ansari, Xi Zhang, Andreas Achtzehn, Marina Petrova, and Petri Mähönen. A Flexible MAC Development Framework for Cognitive Radio Systems. In *IEEE Wireless Communications and Networking Conference (WCNC)*, pages 156–161, Quintana-Roo, Mexico, March 2011.

[104] Vangelis Gazis, Eleni Patouni, Nancy Alonistioti, and Lazaros Merakos. A Survey of Dynamically Adaptable Protocol Stacks. *IEEE Communications Surveys Tutorials*, 12(1):3–23, February 2010.

[105] Gary T. Wong, Matti A. Hiltunen, and Richard D. Schlichting. A Configurable and Extensible Transport Protocol. In *20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 1, pages 319–328, Anchorage, AK, USA, April 2001.

[106] Pierluigi Gallo, Domenico Garlisi, Fabrizio Giuliano, Francesco Gringoli, IIlenia Tinnirello, and Giuseppe Bianchi. Wireless MAC Processor Networking: A Control Architecture for Expressing and Implementing High-Level Adaptation Policies in WLANs. *IEEE Vehicular Technology Magazine*, 8(4):81–89, 2013.

[107] Lars Berlemann, Arnaud Cassaigne, Ralf Pabst, and Bernhard Walke. Modular Link Layer Functions of a Generic Protocol Stack for Future Wireless Networks. In *Software Defined Radio-Technical Conference*, Phoenix, AZ, USA, November 2004.

[108] GNU Radio developers. GNU Radio. Available under `http://gnuradio.org`; last visited on November 11, 2014.

[109] Xi Zhang, Junaid Ansari, Guangwei Yang, and Petri Mähönen. TRUMP: Supporting Efficient Realization of Protocols for Cognitive Radio Networks. In *IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN)*, pages 476–487, Aachen, Germany, May 2011.

[110] Claudia Cormio and Kaushik R. Chowdhury. A Survey on MAC Protocols for Cognitive Radio Networks. *Ad Hoc Networks*, 7(7):1315–1329, September 2009.

[111] Antonio De Domenico, Emilio Calvanese Strinati, and Maria-Gabriella Di Benedetto. A Survey on MAC Strategies for Cognitive Radio Networks. *IEEE Communications Surveys Tutorials*, 14(1):21–44, February 2012.

[112] Mohamed A. Kalil. *Modeling and Analysis of Cognitive Radio Ad Hoc Networks*. PhD thesis, Ilmenau University Of Technology, Ilmenau, Germany, 2011.

[113] Carlos Cordeiro and Kiran Challapali. C-MAC: A Cognitive MAC Protocol for Multi-Channel Wireless Networks. In *2nd IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN)*, pages 147–157, Dublin, Ireland, April 2007.

[114] Ecma International. *Standard ECMA-392: MAC and PHY for Operation in TV White Space*, June 2012.

[115] Kuo-Chun Huang, Xiangpeng Jing, and Dipankar Raychaudhuri. MAC Protocol Adaptation in Cognitive Radio Networks: An Experimental Study. In *18th Internatonal Conference on Computer Communications and Networks (ICCCN)*, San Francisco, CA, USA, August 2009.

[116] Weihong Hu, Homayoun Yousefi'zadeh, and Xiaolong Li. Load Adaptive MAC: A Hybrid MAC Protocol for MIMO SDR MANETs. *IEEE Transactions on Wireless Communications*, 10(11), 2011.

[117] Mark D. Silvius, Allen B. MacKenzie, and Charles W. Bostian. Rendezvous MAC Protocols for Use in Cognitive Radio Networks. In *IEEE Military Communications Conference (MILCOM)*, pages 1–7, Boston, MA, USA, October 2009.

[118] Anna Larmo, Magnus Lindström, Michael Meyer, Ghyslain Pelletier, Johan Torsner, and Henning Wiemann. The LTE Link-Layer Design. *IEEE Communications Magazine*, 47(4):52–59, April 2009.

[119] J. Rajahalme, A. Conta, B. Carpenter, and S. Deering. RFC3697: IPv6 Flow Label Specification, March 2004.

[120] Eddie Kohler, Mark Handley, and Sally Floyd. Designing DCCP: Congestion Control Without Reliability. In *ACM SIGCOMM*, Pisa, Italy, September 2006.

[121] Institute of Electrical and Electronics Engineers. *IEEE Standard for Ethernet 802.3*, December 2012.

[122] Sam Leffler. TDMA for Long Distance Wireless Networks. Technical report, Errno Consulting, 2009.

[123] Ashish Sharma, Mohit Tiwari, and Haitao Zheng. MadMAC: Building a Reconfiguration Radio Testbed using Commodity 802.11 Hardware. In *1st IEEE Workshop on Networking Technologies for Software Defined Radio Networks*, pages 78–83, Reston, VA, USA, September 2006.

[124] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[125] Chris Rupp, Stefan Queins, and Barbara Zengler. *UML 2 glasklar: Praxiswissen für die UML-Modellierung*. Hanser, Munich, Germany, 3. edition, 2007.

[126] Jerry Chun-Ping Wang, Mehran Abolhasan, Daniel R. Franklin, and Farzad Safaei. Characterising the Behaviour of IEEE 802.11 Broadcast Transmissions in Ad Hoc Wireless LANs. In *IEEE International Conference on Communications (ICC)*, pages 1–5, Dresden, Germany, June 2009.

[127] Rajiv D. Banker, Srikant M. Datar, Chris F. Kemerer, and Dani Zweig. Software Complexity and Maintenance Costs. *Communications of the ACM*, 36(11):81–94, November 1993.

[128] Zeeshan Amjad. Mediator Design Pattern Using C++, April 2012. Available under `http://zamjad.wordpress.com/`, last visited on February 26th 2014.

[129] Stuart Kurkowski, Tracy Camp, and Michael Colagrosso. MANET Simulation Studies: The Incredibles. *ACM SIGMOBILE Mobile Computing and Communications Review - Special Issue on Medium Access and Call Admission Control Algorithms for Next Generation Wireless Networks*, 9(4):50–61, October 2005.

[130] Ryan W. Thomas. *Cognitive Networks*. PhD thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, June 2007.

[131] Ettus Research. Application Note Synchronization and MIMO Capability with USRP Devices.
Available under `http://www.ettus.com/content/files/kb/mimo_and_sync_with_usrp.pdf`; last visited on July 2, 2014.

[132] REDHAWK developers. REDHAWK project. Available under `http://redhawksdr.github.io/`; last visited on November 11, 2014.

[133] Joseph D. Gaeddert. liquid-dsp signal processing library. Available under `http://liquidsdr.org/`; last visited on November 11, 2014.

[134] Ahmed Selim, Irene Macaluso, and Linda Doyle. Efficient Sidelobe Suppression for OFDM Systems Using Advanced Cancellation Carriers. In *IEEE International Conference on Communications (ICC)*, pages 4687–4692, Budapest, Hungary, June 2013.

[135] nuttcp developers. nuttcp. Available under `http://www.nuttcp.net/`; last visited on November 5, 2014.

[136] Joseph D. Gaeddert. *Facilitating Wireless Communications through Intelligent Resource Management on Software-Defined Radios in Dynamic Spectrum Environments*. PhD thesis, Virginia Polytechnic Institute & State University, Blacksburg, VA, USA, January 2011.

[137] David G. Messerschmitt. How Digital Communication Works. Technical report, University of California, 1999.

[138] André Puschmann. libgdtp.
Available under `https://github.com/andrepuschmann/libgdtp/`.

[139] Boost developers. Boost C++ Libraries. Available under `http://www.boost.org`; last visited on May 5, 2014.

[140] Apache Software Foundation. log4cxx.
Available under `http://logging.apache.org/log4cxx/`.

[141] The GNU Compiler Collection developers. GCC C++ Coding Conventions. Available under `https://gcc.gnu.org/wiki/CppConventions`; last visited on August 5, 2014.

[142] Free Software Foundation. GNU General Public License v3, June 2007. Available under `http://www.gnu.org/copyleft/gpl.html`; last visited on June 20th 2014.

[143] Richard W. Watson. The Delta-t Transport Protocol: Features and Experience. In *14th Conference on Local Computer Networks (LCN)*, pages 399–407, Minneapolis, Minnesota, USA, October 1989.

[144] Gerrit Renker. Notes on the Implementation of Feature Negotiation in DCCP.
Available under `http://www.erg.abdn.ac.uk/users/gerrit/dccp/notes/feature_negotiation/`; last visited on November 18, 2014.

[145] Thomas Volkert. Homer Conferencing. Available under `http://www.homer-conferencing.com/`; last visited on November 10, 2014.

[146] Google. Protocol Buffers. Available under `http://code.google.com/p/protobuf/`; last visited on December 20th 2014.

[147] Van Jacobson, Robert Braden, and Lixia Zhang. RFC1185: TCP Extension for High-Speed Paths, October 1990.

[148] Deepak R. Joshi, Dimitrie C. Popescu, and Octavia A. Dobre. Gradient-Based Threshold Adaptation for Energy Detector in Cognitive Radio Systems. *IEEE Communications Letters*, 15(1):19–21, January 2011.

[149] Harry Urkowitz. Energy Detection of Unknown Deterministic Signals. *Proceedings of the IEEE*, 55(4):523–531, April 1967.

[150] Romit Roy Choudhury, Abrita Chakravarty, and Tetsuro Ueda. Implicit MAC Acknowledgment: An Improvement to 802.11. In *4th IEEE/ACM Wireless Telecommunications Symposium (WTS)*, pages 1–8, April 2005.

[151] Martin Heusse, Franck Rousseau, Gilles Berger-Sabbatel, and Andrzej Duda. Performance Anomaly of 802.11b. In *22nd Annual Joint Conference of the IEEE Computer and Communications IEEE Societies (INFOCOM)*, pages 1–8, San Francisco, CA, USA, March 2003.

[152] Giuseppe Bianchi, Antonio Di Stefano Costantino G. Giaconi, Luca Scalia Giovanni Terrazzino, and Ilenia Tinnirello. Experimental Assessment of the Backoff Behavior of Commercial IEEE 802.11b Network Cards. In *26th IEEE International Conference on Computer Communications (INFOCOM)*, pages 1181–1189, Anchorage, AK, USA, May 2007.

[153] Giuseppe Bianchi. Performance Analysis of the IEEE 802.11 Distributed Coordination Function. *IEEE Journal on Selected Areas in Communications*, 18(3):535–547, March 2000.

[154] Andrzej Duda. Understanding the Performance of 802.11 Networks. In *IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1–6, Cannes, France, September 2008.

[155] J. Colman O'Sullivan, Paolo Di Francesco, Uchenna K. Anyanwu, Luiz A. DaSilva, and Allen B. MacKenzie. Multi-hop MAC Implementations for Affordable SDR Hardware. In *IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN)*, pages 632–636, Aachen, Germany, May 2011.

[156] Peter Dely, Andreas J. Kassler, and Dmitry Sivchenko. Theoretical and Experimental Analysis of the Channel Busy Fraction in IEEE 802.11. In *Future Network and Mobile Summit*, pages 1–9, Florence, Italy, June 2010.

[157] Ameer Ahmed Abbasi and Mohamed Younis. A Survey on Clustering Algorithms for Wireless Sensor Networks. *Computer Communications*, 30(14-15):2826–2841, October 2007.

[158] Yi Song and Jiang Xie. Common Hopping Based Proactive Spectrum Handoff in Cognitive Radio Ad Hoc Networks. In *IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1–5, Miami, FL, USA, December 2010.

[159] Fei Hu Yang Xiao, editor. *Cognitive Radio Networks*. Auerbach Publications, December 2008.

[160] Miguel López-Benítez and Fernando Casadevall. An Overview of Spectrum Occupancy Models for Cognitive Radio Networks. In *IFIP TC 6th International Conference on Networking*, pages 32–41, Valencia, Spain, May 2011.

[161] Stefan Geirhofer, Lang Tong, and Brian M. Sadler. Dynamic Spectrum Access in WLAN Channels: Empirical Model and Its Stochastic Analysis. In *First International Workshop on Technology and Policy for Accessing Spectrum (TAPAS)*, New York, NY, USA, August 2006.

[162] Miguel López-Benítez and Fernando Casadevall. Time-Dimension Models of Spectrum Usage for the Analysis, Design, and Simulation of Cognitive Radio Networks. *IEEE Transactions on Vehicular Technology*, 62(5):2091–2104, January 2013.

[163] Puneet Sharma, Deborah Estrin, Sally Floyd, and Van Jacobson. Scalable Timers for Soft State Protocols. In *16th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 1, pages 222–229, Kobe, Japan, April 1997.

[164] Ping Ji, Zihui Ge, Jim Kurose, and Don Towsley. A Comparison of Hard-state and Soft-state Signaling Protocols. *IEEE/ACM Transactions on Networking (TON)*, 15(2):281–294, April 2007.

[165] Maziar Nekovee. A Survey of Cognitive Radio Access to TV White Spaces. *International Journal of Digital Multimedia Broadcasting*, pages 1–11, April 2010.

[166] Hyoil Kim and Kang G. Shin. In-band Spectrum Sensing in Cognitive Radio Networks: Energy Detection or Feature Detection? In *14th ACM International Conference on Mobile Computing and Networking (MobiCom)*, pages 14–25, San Francisco, CA, USA, September 2008.

[167] IEEE Computer Society. *IEEE 802 Standard Part 22: Cognitive Wireless RAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Policies and Procedures for Operation in the TV Bands*. Institute of Electrical and Electronics Engineers, July 2011.

[168] André Puschmann. MobilitySim.
Available under `https://github.com/andrepuschmann/mobilitysim/`.

[169] André Puschmann. RendezvousSim.
Available under `https://github.com/andrepuschmann/rendezvoussim/`.

[170] Luiz A. DaSilva and Igor Guerreiro. Sequence-Based Rendezvous for Dynamic Spectrum Access. In *3rd IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN)*, pages 1–7, Chicago, IL, USA, October 2008.

[171] D. Yang, J. Shin, and C. Kim. Deterministic Rendezvous Scheme in Multichannel Access Networks. *Electronics Letters*, 46(20):1402–1404, September 2010.

[172] Gábor Fodor, Erik Dahlman, Gunnar Mildh, Stefan Parkvall, Norbert Reider, György Miklós, and Zoltán Turányi. Design Aspects of Network Assisted Device-to-Device Communications. *IEEE Communications Magazine*, 50(3):170–177, March 2012.

Erklärung

Ich versichere, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Weitere Personen waren an der inhaltlich-materiellen Erstellung der vorliegenden Arbeit nicht beteiligt. Insbesondere habe ich hierfür nicht die entgeltliche Hilfe von Vermittlungs- bzw. Beratungsdiensten (Promotionsberater oder anderer Personen) in Anspruch genommen. Niemand hat von mir unmittelbar oder mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen. Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form einer Prüfungsbehörde vorgelegt. Ich bin darauf hingewiesen worden, dass die Unrichtigkeit der vorstehenden Erklärung als Täuschungsversuch bewertet wird und gemäß §7 Abs. 10 der Promotionsordnung den Abbruch des Promotionsverfahrens zur Folge hat.

Ort, Datum	André Puschmann