AN ARCHITECTURE FOR CONVERGING RECONFIGURABLE RADIO SYSTEMS

Ryan Michael van den Bergh

A thesis submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Doctor of Philosophy.

Johannesburg, 2012

Declaration

I declare that this thesis is my own, unaided work, other than where specifically acknowledged. It is being submitted for the degree of Doctor of Philosophy in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other university.

Signed this _____ day of _____ 2012

_____

Ryan Michael van den Bergh

Abstract

Since mobile telecommunication systems were first introduced in the early 1980s they have become a pervasive part of modern life, with an estimated 85% of the global population believed to be in possession of a mobile communications device. To address the ever-increasing demand for fast ubiquitous provision of multimedia and data services, new Radio Access Technologies (RATs) capable of meeting those demands are constantly being developed and standardised. Currently the fourth generation of RATs is being deployed by network operators around the world, with standards bodies already working to develop and standardise even more advanced RATs.

The introduction of any new, and often upgraded, RATs almost always requires network operators to purchase new hardware systems capable of supporting the new RATs, which must then be integrated with the plethora of RATs already present in the network operator's heterogeneous Radio Access Network (RAN). This process is costly and poses risks for network operators, as they must first invest significant amounts of capital on new network hardware and then they have to convince their subscribers to purchase new mobile devices which are capable of supporting the new RAT. Reconfigurable Radio Systems (RRSs) are a relatively new approach to developing, implementing and managing RATs within a RAN. A RRS differs from a traditional radio system, in that each RAT is defined in software which can be reused across multiple generic hardware platforms. Many RRSs also provide the functionality to manage and control the dynamic implementation of different RATs in network elements throughout a RAN.

Although RRSs are the subject of numerous research efforts, there is currently no unifying approach or set of requirements for an RRS architecture or framework. Instead various researchers focus their efforts on specific topics relating to RRS, such as the reconfigurable management system, or how RATs are modelled and implemented in software. This lack of formal standardisation or approach to developing RRSs represents a hindrance to the widespread adoption of RRSs.

After having reviewed and analysed the architectures and functionality of all major RRSs developed by researchers and standardisation initiatives, this thesis defines a set of combined functional requirements which forms the basis for the development of a Converged RRS Architecture (CRRSA) capable of both managing and controlling the reconfiguration operations of a RAN, and dynamically implementing any RAT at runtime. The CRRSA was developed using a greenfield approach, thereby avoiding the limitations of existing RRSs, which cannot be simply upgraded or modified to support all of the defined functional requirements, both for practical reasons and due to architectural and technological limitations.

The CRRSA is an extremely flexible and scalable architecture which defines a series of technology-independent logical components and interfaces for managing and performing reconfiguration operations in the Subscriber, Access Network and Core Network domains. The CRRSA is unique in that it provides the capability to customise and upgrade its reconfiguration management system through the use of logical components called Reconfigurable Algorithms.

The implementation of RATs in the CRRSA is achieved by decomposing each RAT into its fundamental components, and encapsulating the resulting functions within software components called Reconfigurable Components, which are then used to construct the RAT during reconfiguration operations. The primary benefit of this methodology is that any RAT can be implemented simply by choosing and connecting the appropriate Reconfigurable Components.

In order to prove the CRRSA's practical viability, the CRRSA was implemented and tested in a series of simulations using two RAT models derived from the IEEE 802.16 and 3GPP LTE standards. The results of these simulations confirmed that the CRRSA is capable of managing and controlling all reconfiguration operations throughout a RAN, and that it is capable of practically implementing RATs via Reconfigurable Components, which are not only able to successfully transmit signals in a simulated environment, but which are also able to transmit and receive signals carrying data through the wireless environment using an actual SDR hardware platform.

The CRRSA thus provides a key contribution to the field of RRS by delivering a detailed unified logical architecture which can be used to develop and standardise RRSs, and which possesses flexible and scalable methods for reconfiguring and adapting the functionality of its reconfigurable management system and RATs in a real wireless environment using SDR hardware platforms.

Acknowledgements

I would like to thank my supervisor, Prof. Hu Hanrahan, for his knowledge, guidance and support throughout the duration of this research. I would also like to thank Prof. Rex van Olst for his encouragement and advice. A special thanks goes to my friends and colleagues at the Centre for Telecommunications Access and Services (CeTAS) at the University of the Witwatersrand for their support and advice, especially Doron Horwitz, Mehroze Abdullah, David Vannucci, Rolan Christian and Folasade Dahunsi.

Finally, I would like to thank my parents, Mike and Wendy van den Bergh, my sister, Kirsty van den Bergh and my girlfriend, Helen Wright, for their love, support, guidance, and patience without which I could never have completed this research.

This work was performed at the Centre for Telecommunications Access and Services, at the University of the Witwatersrand, Johannesburg. The Centre is funded by Telkom SA Limited, Nokia-Siemens Networks, Telsaf Data, and the Department of Trade and Industry's THRIP programme. Their financial support is greatly appreciated.

Contents

List of Figures

List of Tables

List of Abbreviations

1G	First Generation
2G	Second Generation
3G	Third Generation
3GPP	Third Generation Partnership Project
3GPP2	Third Generation Partnership Project 2
4G	Fourth Generation
ACL	Agent Communication Language
ADC	Analogue to Digital Converter
AEP	Application Environment Profile
AMPS	Advanced Mobile Phone System
API	Application Programming Interface
ARQ	Automatic Repeat Request
ATM	Asynchronous Transfer Mode
AWGN	Additive White Gaussian Noise
CAPEX	Capital Expenditure
CCM	Configuration Control Module
CDMA	Code Division Multiple Access
CF	Core Framework
CID	Connection Identifier
CORBA	Common Object Request Broker Architecture
COTS	Commercial off-the-shelf
CPC	Cognitive Pilot Channel
CPRI	Common Public Radio Interface
CPS	Common Part Sublayer
CR	Cognitive Radio
CP	Cyclic Prefix
CRC	Cyclic Redundancy Check
CRRSA	Converged Reconfigurable Radio System Architecture
CS	Service-Specific Convergence Sublayer

DAC	Digital to Analogue Converter
DAMPS	Digital Advanced Mobile Phone System
DFT	Digital Fourier Transform
DLL	Data Link Layer
DL-SCH	Downlink Shared Channel
DM	Decision Making
DSA	Dynamic Spectrum Access / Allocation
DSM	Dynamic Spectrum Management
DSONPM	Dynamic Self-Organising Network Planning and Management
DSP	Digital Signal Processor
DTD	Document Type Definition
E^{2}R	End-to-End Reconfigurability
E^{2}RII	End-to-End Reconfigurability 2
E3	End-to-End Efficiency
EDGE	Enhanced Data Rate for GSM
EPC	Evolved Packet Core
ETSI	European Telecommunications Standards Institute
E-UTRAN	Evolved UTRAN
EV-DO	Evolution-Data Optimized
EV-DV	Evolution-Data and Voice
FA	Functional Architecture
FDD	Frequency Division Duplexing
FDM	Frequency Division Multiplexing
FEC	Forward Error Correction
FIPA	Foundation for Intelligent Physical Agents
FPGA	Field Programmable Gate Array
GPP	General Purpose Processor
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
HARQ	Hybrid Automatic Repeat Request
HSDPA	High-speed Downlink Packet Access
HSUPA	High-speed Uplink Packet Access
ICI	Inter-carrier Interference
IDL	Interface Description Language
IEEE	Institute of Electrical and Electronics Engineers
IFFT	Inverse Fast Fourier Transform

IMT	International Mobile Telecommunication
IP	Internet Protocol
ISI	Inter-symbol Interference
ITU	International Telecommunications Union
JADE	Java Agent Development Framework
JAR	Java Archive
JNI	Java Native Interface
JPEO	Joint Program Executive Office
JRRM	Joint Radio Resource Management
JTRS	Joint Tactical Radio System
JVM	Java Virtual Machine
LTE	Long Term Evolution
MAC	Medium Access Control
NE	Network Equipment
NMT	Nordic Mobile Telephone
NRM	Network Reconfiguration Manager
NS	Network Simulator
OBSAI	Open Base Station Architecture Initiative
OFDM	Orthogonal Frequency Division Multiplexing
OFDMA	Orthogonal Frequency Division Multiple Access
OMG	Object Management Group
OPEX	Operational Expenditure
OSI	Open Systems Interconnection
OSSIE	Open Source SCA Implementation Embedded
OTA	Over-the-air
PAPR	Peak to Average Power Ratio
PDC	Personal Digital Cellular
PDCP	Packet Data Convergence Protocol
PDSC	Packet Downlink Shared Channel
PDU	Packet Data Unit
PHC	Packet Header Compression
PIM	Platform Independent Model
PMP	Point-to-Point
PMUX	Packet Multiplexer
POSIX	Portable Operating System Interface
PSM	Platform Specific Model
PUSC	Packet Uplink Shared Channel
QAM	Quadrature Amplitude Modulation

QoS	Quality of Service
QPSK	Quadrature Phase Shift Keying
RAN	Radio Access Network
RAT	Radio Access Technology
RBS	Reconfigurable Base Station
RC	Reconfigurable Controller
RCAL	Reconfigurable Component and Algorithm Library
RE	Radio Equipment
REC	Radio Equipment Control
RF	Radio Frequency
RLC	Radio Link Control
RMD	Reconfigurable Mobile Device
RMM	Reconfigurable Monitoring Module
RPM	Reconfigurable Policy Manager
RPS	Reconfigurable Protocol Stack
RRRM	Reconfigurable Radio Resource Manager
RRS	Reconfigurable Radio System
RSSI	Received Signal Strength Indicator
SAX	Simple API for XML
SCA	Software Communications Architecture
SCC41	IEEE Standards Coordinating Committee 41
SC-FDMA	Single Carrier Frequency Division Multiple Access
SCM	Secure Communications Module
SDH	Synchronous Digital Hierarchy
SDR	Software Defined Radio
SDU	Service Data Unit
SFID	Service Flow Identifier
SNR	Signal to Noise Ratio
SoC	System on a Chip
SOFDM	Scalable OFDM
SONET	Synchronous Optical Network
TDD	Time Division Duplexing
TRM	Terminal Reconfiguration Manager
UHD	USRP Hardware Driver
UL-SCH	Uplink Shared Channel
UML	Unified Modelling Language
UMTS	Universal Mobile Telephone Service
UTRAN	Universal Terrestrial Radio Access Network

USRP	Universal Serial Radio Peripheral
W3C	World Wide Web Consortium
W-CDMA	Wideband CDMA
WiMAX	Worldwide Interoperability for Microwave Access
WINNER	Wireless World Initiative New Radio
WMUX	Waveform Multiplexer
XML	Extensible Markup Language

Chapter 1

The Evolution and Standardisation of Mobile Telecommunication Systems

Mobile telecommunication systems have become an integral part of modern life, enabling billions of people worldwide to perform their daily activities, conduct business, and keep in contact with friends and family. Currently around 85% of the global population is estimated to be in possession of a mobile communication device, representing a total of 5.88 billion connections [1].

Numerous wireless and mobile telecommunication standards have been developed by a variety of telecommunication standards organisations. The need to provide new and innovative services has resulted in standards organisations, network operators and equipment vendors constantly developing new Radio Access Technologies (RAT), each of which must be integrated into increasingly complex heterogeneous Radio Access Networks (RAN).

The introduction of each new RAT requires that network operators and subscribers purchase the required network equipment to support the RAT, which is both costly and wasteful [2, 3]. Reconfigurable Radio Systems (RRS) are a relatively recent concept which is being touted as the solution to this problem. RRSs abstract RAT functionality from the underlying hardware, by defining such functionality in software and standardising a common hardware platform [2–6]. This means that new RATs can be deployed on existing hardware, saving network operators and subscribers from having to continuously invest in costly hardware upgrades. A number of additional benefits relating to the joint management of radio resources and efficient use of radio spectrum are also provided by RRSs [2–4, 7].

The following sections provide additional insight into the history and evolution of mobile telecommunication systems and networks, a discussion regarding the process by which RAT standards are currently developed, and an overview of Reconfigurable Radio System's research and challenges. Based on the presented information, this Thesis's objectives, criteria, and constraints are then defined, followed by an outline of the topics covered in the remaining chapters.

1.1 The History and Evolution of Mobile Telecommunications

Each generation of mobile communication systems has been characterised by a significant increase both in the functionality provided to the user and the level of technology required to provide new services [8, 9]. Since the first mobile communications systems appeared in the late 1970s, four generations of RATs have been developed and standardised.

Figure 1.1 illustrates the evolution and standardisation of all major RAT standards, and provides a reference for the following subsections which examine the history and functionality provided by each generation of mobile telecommunications systems.

1.1.1 First Generation Mobile Telecommunication Systems

The first generation (1G) of mobile telecommunications systems was introduced in the late 1970s and provided basic mobile telephony services [11, 13, 14, 16]. These systems made use of analogue based circuitry, which resulted in extremely poor quality of service and limited capacity within each cell [11, 14, 17].

To complicate matters further, the researchers and standards organisations in each geographical region developed their own RAT standard, which made international roaming impossible [12, 14]. The majority of these standards were developed in one of three main regions: North America, Europe, and Japan.

Examples of 1G RAT standards include: the MCS-L1 system from Japan, the Advanced Mobile Phone System (AMPS) from the US and the Nordic Mobile Telephone (NMT) system from Europe. Table 1.1 contains the key specifications for each of these standards. Please note that the list of assigned frequency bands

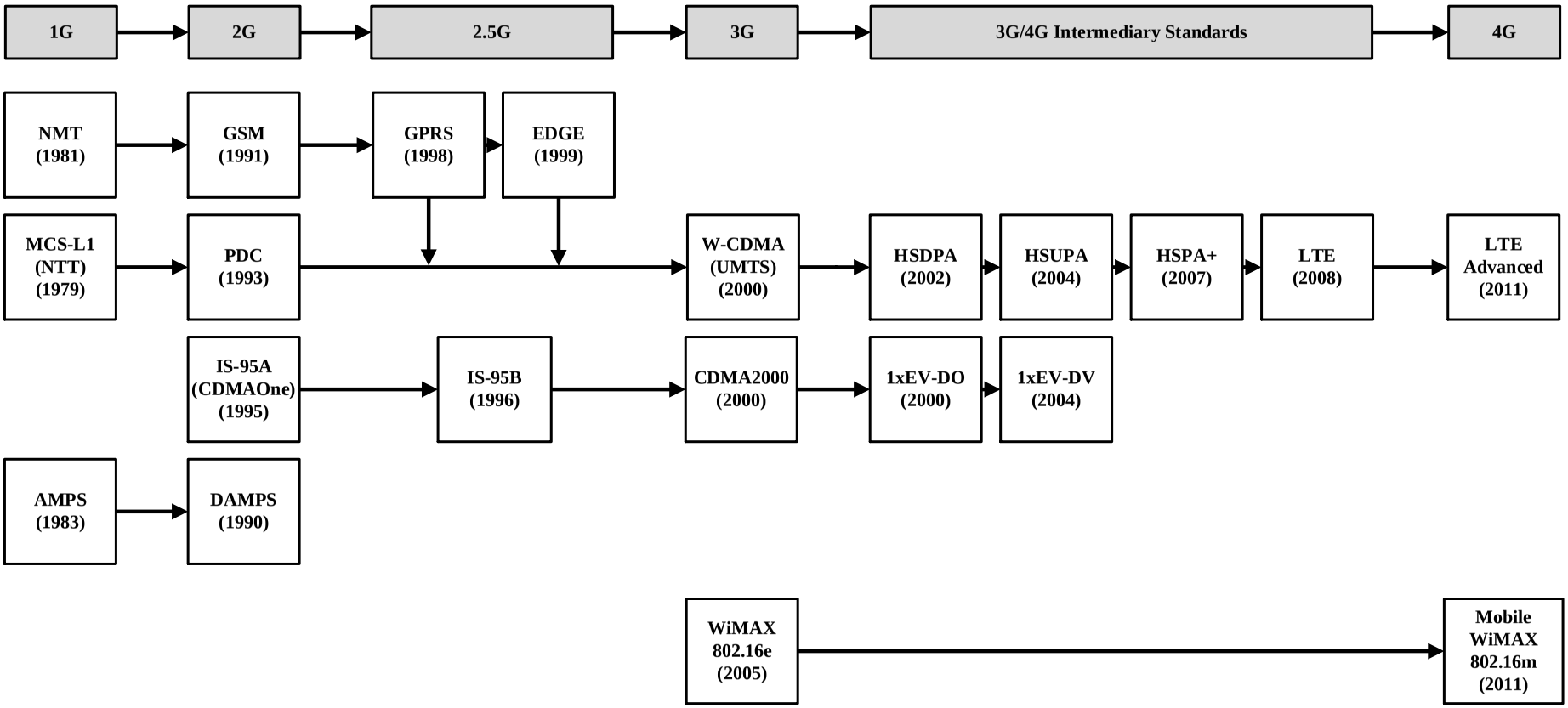


Figure 1.1: Evolution of Mobile Telecommunication Standards [9–16]

in Table 1.1 is not exhaustive, and is presented as an indication of the spectrum requirements for each 1G RAT.

Table 1.1: 1G RAT standard specifications and characteristics [11, 13, 15, 16]

	AMPS	**NMT450**	**MCS-L1 (NTT)**
Region	North America	Europe	Japan
Downlink Spectrum (MHz)	824-849	453-457.5	925-940
Uplink Spectrum (MHz)	869-894	463-467.5	870-885
Channel Bandwidth (kHz)	30	25	25
Modulation	FM	FM	FM
Multiple Access	FDMA	FDMA	FDMA

1.1.2 Second Generation Mobile Telecommunication Systems

The second generation (2G) of mobile telecommunication systems were introduced in the early 1990s, and signalled the start of a mobile communications revolution that has since swept across the globe [10, 11, 14].

2G systems replaced the first generation's legacy analogue components with digital counterparts. This development enabled the use of digital modulation schemes in combination with channel coding algorithms and resulted in a substantial improvement in quality of service and network capacity over 1G systems [11].

2G systems were, and still are, immensely popular with mobile telephony users worldwide due to their simplicity and relatively low cost. Perhaps the most well-known, and still the most widely adopted, 2G RAT is the Global System for Mobile Communications (GSM) [9]. The GSM standard, initially developed in Europe, has since been deployed globally, with over 3 billion GSM mobile devices currently in use [18].

Some other examples of 2G RATs, whose initial specifications along with GSM's are listed in Table 1.2, are: the Digital AMPS (DAMPS) and IS-95A systems deployed in the US, and the Japanese Personal Digital Cellular (PDC) system.

2G systems were initially developed to provide voice services, however the advent and subsequent wide-spread adoption of the Internet in the 1990s forced researchers and standards bodies to examine possible methods by which the GSM standard could be extended to provide mobile data services. Unfortunately the circuit switched nature of 2G technology makes it extremely inefficient for providing data services [11, 13].

Table 1.2: 2G RAT standard specifications and characteristics [11, 13, 15, 19]

	DAMPS	**GSM**	**IS-95 (CDMAOne)**	**PDC**
Region	North America	International (Initially Europe)	North America	Japan
Channel Bandwidth (kHz)	30	200	1250	25
Modulation	$\pi/4$ DQPSK	BPSK/OQPSK	GMSK	$\pi/4$ DQPSK
Multiple Access	TDMA	TDMA	CDMA	TDMA
Speech Codec	VSELP: 8kbps	RPELPC: 13 kbps VSELP: 8 kbps	EVRC: 13 kbps CELP: 8kbps	VSELP: 9.6 kbps CELP 5.6 kbps

To resolve this issue, researchers modified the 2G air-interface, and a packet-switched network was overlaid and interconnected with the existing 2G circuit-switched systems [11, 19]. Thus voice services continued to be provided via the pre-existing circuit-switched core network with the original air-interface, while data services utilised the packet-switched core network in combination with a modified 2G air-interface containing advanced modulation and adaptive coding [11, 13, 15, 19]. The exact throughput for each system is dependent on the modulation scheme and number of time slots (i.e. voice channels) allocated to each data channel.

Although 2.5G systems provide data service provisioning capability, they did not provide sufficient additional capabilities to be classified as a new generation of mobile systems in their own right. As such, these systems are colloquially referred to as 2.5G or pre-3G systems. Examples of 2.5G systems include: the General Packet Radio Service (GPRS) and the Enhanced Data Rates for GSM (EDGE) systems and the IS-95B system for IS-95A systems [10, 11, 13, 15, 19].

1.1.3 Third Generation Mobile Telecommunication Systems

In the late 1990s the International Telecommunications Union (ITU) introduced its plan, known as the International Mobile Telecommunications Vision or IMT-2000, to unify all wireless communications standards under one single global standard and within the same set of frequency bands [9, 11, 13, 15, 17].

The final IMT-2000 standard, introduced in the early part of the 21st century, encompasses a satellite specification standard and multiple cellular air-interface architecture standards, all connected with a single packet switched core network and capable of providing multimedia services [9, 12, 13]. The cellular network component of the IMT-2000 standard is referred to as the Third Generation (3G) of mobile communication systems.

It should be noted that, despite the ITU's efforts, its vision of a single unifying global cellular standard was never realised [11,13]. Many of the original 1G and 2G regional standardisation bodies had merged or evolved into larger organisations, resulting in two main international cellular standards bodies, namely the Third Generation Partnership Project (3GPP) [20], and the Third Generation Partnership Project 2 (3GPP2) [21].

Figure 1.1 illustrates the fact that, despite the plethora of 1G and 2G standards in existence, only two 3G standards were widely adopted when IMT-2000 was initially launched, namely, the Wideband Code Division Multiple Access (W-CDMA) system developed as part of the Universal Mobile Telephone Service (UMTS) and proposed and maintained by the 3GPP, and the CDMA2000 system proposed and maintained by the 3GPP2 [9, 11–13, 15].

The Institute of Electrical and Electronic Engineers (IEEE) began development of the 802.16 standard series during the late 1990s [15, 22, 23]. The 802.16 standard, also known as Worldwide Interoperability for Microwave Access or WiMAX, enables the development and deployment of systems which are similar to Wi-Fi systems in the sense that they can be used to provide users with cheap high speed access to Internet services; however the underlying technologies are vastly different and the 802.16 cell size is considerably larger than Wi-Fi (usually several kilometres) [22–24]. Although 802.16 was originally developed as a wireless networking standard, in 2007 at the request of the IEEE the ITU formally included the 802.16 standard within the IMT-2000 framework, thereby classifying 802.16 as a 3G RAT [25]. Table 1.3 contains some of the key specifications for the W-CDMA, CDMA2000, and 802.16 3G RATs, and Chapter 4 examines the mobile version of the 802.16 standard (802.16e) in greater detail.

Table 1.3: 3G RAT standard specifications and characteristics [11,12,15,22–24]

	W-CDMA (UMTS)	**CDMA2000**	**802.16e (Mobile WiMAX)**
Standards Body	3GPP	3GPP2	IEEE
Maximum Data Rate	384 kbps	153 kbps	46 Mbps
Channel Coding	Convolutional and Turbo Coding	Convolutional and Turbo Coding	Convolutional and Turbo Coding
Modulation	BPSK and QPSK	BPSK and QPSK	QPSK, 16QAM, and 64QAM
Multiple Access	CDMA	CDMA	OFDMA

While 3G systems provide a significantly higher data throughput as opposed to 2G and 2.5G systems, and are also capable of providing high performance data and

multimedia services, the deployment of 3G networks and systems was fraught with delays and slow user adoption rates. Although the effects of high implementation costs and exorbitant spectrum license fees contributed to this situation, the primary reason for the slow adoption of 3G systems was users' attitudes towards the services provided by 3G network operators [14, 26–28]. While 3G systems were developed with a set of benchmark services intended to meet users' expectations, mobile network operators found that many users did not want to invest in new 3G terminal equipment purely in order to access services that in their own right were not considered sufficiently novel or compelling. As a result 3G network operators initially experienced difficulties in generating a return on their 3G system capital expenditure.

1.1.4 Beyond 3G and Fourth Generation Mobile Telecommunication Systems

In the years immediately following the initial deployment of 3G systems in 2001, many theories regarding the nature and characteristics of the next generation of mobile communications systems were presented by both researchers and standards bodies.

The ITU attempted to take a leading role in the development of a Fourth Generation (4G) mobile communications standard, much in the same way that it did with the IMT-2000 framework during the process of standardizing 3G systems, through the publication of Recommendation M.1645 [29] in 2003. In this recommendation the ITU attempted to combine the two leading theories regarding 4G standards development, namely the evolution and convergence of existing RAT, into a single universal RAT standard, and the development of new RAT standards designed to satisfy users' demands for advanced services.

Unfortunately this vision was never completely realised, as the standards bodies responsible for the development of 3G standards chose to focus their efforts on making additions to existing 3G air-interface architectures, rather than converging all existing RATs into a single architecture which would have led to a new converged 4G standard. By including new signal processing algorithms and technologies, standards bodies were able to improve the performance of existing 3G systems with only minimal changes to the 3G standards [12, 13, 15, 17, 19].

As these systems are considered to be evolved versions of 3G systems, they are colloquially referred to as 3.5G or 3G/4G intermediary systems. Examples of such standards include the High-Speed Downlink Packet Access (HSDPA) and High-Speed Uplink Packet Access (HSUPA) systems which upgrade the performance of W-CDMA 3G systems and the Evolution-Data Optimized(EV-DO) and Evolution-Data and Voice (EV-DV) systems which upgrade the performance of CDMA2000 3G systems [12, 13, 15, 17, 19, 22].

Although 3.5G RATs provided a significant performance increase over 3G systems, the advent of smartphones and their associated mobile applications has generated increased demand for high performance RATs capable of supporting such technologies. As the 3.5G RATs were based on the same core technologies contained in 3G systems, standards bodies could no longer perform simple upgrades on existing systems but instead were forced into developing entirely new RATs based on new approaches and signal processing technologies. The advantage of this situation is that standards bodies can develop new RATs without the limitations of, or requirement to support, legacy technologies and systems. The disadvantage however, is that network operators are required to spend significant amounts of their capital purchasing new network equipment and constructing new RANs.

In 2008 the 3GPP published the first version of their Long Term Evolution (LTE) standard [30–32]. The LTE specification consists of two parts, the evolved access network, referred to as the Evolved Universal Terrestrial RAN (E-UTRAN), and the evolved core network, referred to as the Evolved Packet Core (EPC) [23, 31]. Unlike 3G, or even 3.5 systems, LTE contains only a packet-based core network, making it ideally suited to providing advanced multimedia and data services. The quality of service control mechanisms, improved spectral efficiency, efficient management of radio resources, and flexible bandwidth allocation are just some of the key functions that differentiate LTE from previous RATs [17, 23, 30, 31].

To improve the performance of LTE beyond that of previous RATs, the 3GPP chose to incorporate signal processing technologies that had not previously been used in 3G and 3.5 systems. Examples of such technologies include multiple transmitting and receiving antennas, advanced digital modulation schemes such as Orthogonal Frequency Division Multiplexing (OFDM), and a new multiple access scheme based on OFDM called Orthogonal Frequency Division Multiple Access (OFDMA) [17, 23, 30, 31]. Although these technologies are responsible for LTE's increased performance, their inclusion within the standard also results in network operators and subscribers needing to purchase new network equipment. A more detailed explanation of the LTE standard and its capabilities is presented in Chapter 4.

There have been many conflicting opinions regarding whether LTE should be classified as a 3.5G or 4G standard, largely due to the lack of a standardised definition for what constitutes a 4G RAT [33]. Many network operators have marketed their LTE networks, and even their WiMAX networks, as 4G systems in an attempt to attract new subscribers and increase their market share [33].

In 2008 the ITU issued a set of requirements for a new set of RAT standards known as IMT-Advanced [34,35]. The IMT-Advanced framework is a successor to the IMT-2000 framework, which contained all of the 3G RAT standards. The requirements published by the ITU vastly exceeded the capabilities of all existing systems at the time. As a result, the 3GPP and IEEE both initiated projects to improve their LTE and 802.16 standards, in order to meet the ITU's IMT-Advanced requirements. Both of these standards bodies published new versions of their RATs in 2011, named LTE-Advanced [17,19,30,32] and WiMAX Advanced (802.16m) [12,23], which were subsequently submitted and accepted by the ITU for inclusion within the IMT-Advanced framework [36].

IMT-2000 provided the framework for specifying the requirements of all 3G systems, and therefore it follows that IMT-Advanced is responsible for specifying the requirements for all 4G systems. Due to the fact that neither LTE, nor any prior RAT, met the requirements specified by IMT-Advanced, many researchers, standards bodies, and even some commercial entities have indicated their belief that only LTE-Advanced and WiMAX Advanced (802.16m) could be classified as 4G systems, provided they are accepted by the ITU for inclusion within the IMT-Advanced framework [33]. The ITU provided some clarification at the end of 2010, when they stated that the term 4G, "while undefined, may also be applied to the forerunners of these technologies, LTE and WiMAX, and to other evolved 3G technologies providing a substantial level of improvement in performance and capabilities with respect to the initial third generation systems now deployed" [37].

The previous subsections provided a brief overview of the history and evolution of mobile telecommunication systems. The information provided is intended to provide context for the discussion in the next section, regarding the development and standardisation of RATs.

1.2 Development and Standardisation of Radio Access Technologies

Modern mobile telecommunications contain many different types of mobile and fixed-line access technologies. It is the task of a network operator to integrate all of these systems into a single heterogeneous communications network. Depending on the number and type of access technologies involved, this can prove to be a very complex and costly exercise.

Figure 1.2 illustrates two of the key differences between different access network technologies, namely mobility and throughput. This diagram, based on the ITU vision for a converged 4G standard in Recommendation M.1645 [29], also demonstrates the number and variety of access network technologies that are currently standardised and deployed worldwide.

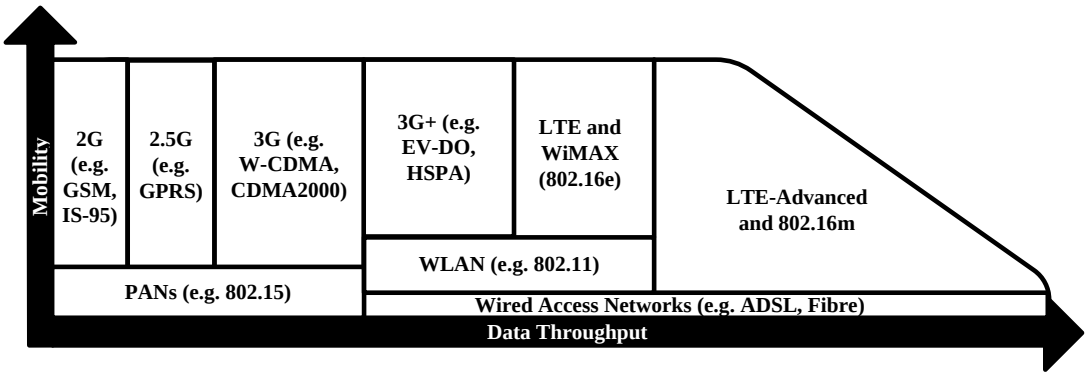


Figure 1.2: Comparison of the throughput and mobility provided by access technologies [12, 23, 29]

While the first and second generations of mobile communication systems were only focused on providing telephony, subsequent generations have broadened the scope of mobile services to include data and multimedia. A network operator must therefore find solutions for the integration and management of RATs, such that the most appropriate RAT is used to provide services to each subscriber.

The introduction of smartphones and social networking applications has dramatically increased the demand for personalised high performance mobile services, resulting in mobile communication standards bodies developing new RATs, containing new

communication protocols and signal processing techniques, in order to provide such services. This RAT development cycle, whereby users' demand for services results in the development and standardisation of new RATs, is expected to continue for the foreseeable future [16, 17].

As each new RAT is standardised, implemented, and then deployed, network operators must first purchase and integrate the new RAT's network equipment into existing networks, and then encourage subscribers to purchase mobile devices in order to avail of the new RAT's capabilities. This means that not only are network operators required to spend large amounts of capital in acquiring and integrating new network equipment, but their ability to recover this initial capital expenditure is based largely on whether they can convince their subscribers that the capabilities offered by each new RAT justifies the cost of a new mobile device.

This particular problem was highlighted during the years immediately following the deployment of 3G RATs. Many network operators spent significant amounts of capital on the network equipment and at auctions for the relevant spectrum access licenses, and were subsequently forced to incur huge financial losses as many subscribers did not see the benefit of purchasing new 3G mobile devices as their 2G and 2.5 mobile devices provided all the voice and data services they required [14, 26–28]. While the industry has to a large degree learnt from this experience, and requirements for new RATs are more carefully analysed to ensure that new systems provide services that are attractive to subscribers, it is not possible to guarantee that new RATs will be financially successful.

Large standards bodies, such as the 3GPP or IEEE, tend to take considerable amounts of time when developing and standardising new RATs. While this is in part due to the scope and complexity of each standard, the committee-based standardisation process used by most large standards bodies results in lengthy consultations concerning the choice of RAT requirements, architecture, and incorporated technologies [17, 38].

This committee-based approach also suffers from the disadvantage that government regulatory bodies, large network operators, service competitors, equipment vendors, and other parties affiliated to the standards body can use the process to ensure that their own interests are served when developing each RAT standard [16, 39]. This means that each RAT may not incorporate the most appropriate or modern technologies, and it also doesn't necessarily possess all of the functionality desired by parties who are not involved in the standardisation process.

Most major RAT standards also contain components whose functionality is not necessarily well-defined. These "holes" in the standard are usually as a result of such functionality being considered outside of the scope of the standard, which could in turn be the result of a committee failing to reach a decision. Nevertheless such "holes" must be resolved when RAT standards are implemented, thereby requiring equipment vendors to implement their own proprietary solutions, which can have a negative impact on the interoperability of system defined by the same standard but implemented by different vendors.

To summarise, the current RAT development and standardisation process, while responsible for the creation of all existing RAT systems, possesses a number of key financial disadvantages for both network operators and subscribers. In addition, the process itself is often lengthy and does not always produce an optimal result. The following subsection presents a relatively new and revolutionary approach to developing, deploying, and managing RATs within a mobile communications network, and which is capable of eliminating the problems with the current RAT development and standardisation process.

1.3 Reconfigurable Radio Systems

Each RAT standard provides different functionality and contains different signal processing and communications technologies. This is the primary reason why network operators and subscribers are forced to purchase new network equipment whenever a new RAT standard is released. Researchers as early as the 1970s realised that if such functionality was defined in software and thereby separated from its underlying hardware, then such systems would be capable of reusing existing hardware when implementing different RATs [2, 3, 6].

It is this principle which forms the basis for a new approach to the development, implementation, and management of RATs, an approach known as Reconfigurable Radio Systems (RRS). RRS is a generic term used by researchers to define radio systems which implement their functionality in software, together with operations involved in the management and control of such systems [2–5, 7]. Due to RRSs' extreme signal processing requirements, practical RRS and associated technologies have only been available since the early-to-mid 1990s [2,6]. The use of RRSs provides a number of advantages over classic communication systems [2–5, 7]:

- Pre-existing RRS compatible hardware can be reused to implement, deploy, and manage new RAT standards as they are released. This minimises the capital expenditure required to deploy and manage new RATs.
- New standards can be deployed with significantly decreased financial risk, as network operators can eliminate or replace any RATs which are not producing a significant financial return.
- New RATs can be rapidly deployed to all network elements through the use of secure software download mechanisms.
- Numerous RATs can be deployed on the same hardware, thereby simplifying integration of RATs within a heterogeneous access network.
- Development of new RATs is cheaper and faster as it is performed in software, thereby providing developers with the ability to make use of pre-existing software components written for other RATs.
- The cost and management effort required for all network components is decreased due to the increased flexibility resulting from all functionality being defined in software.
- The radio resources of a RAN can be jointly managed, enabling a network operator to optimise the use of such resources and dynamically adjust the capacity of the network based on the current load.
- More efficient use of the spectrum can be achieved, as the frequency bands can be dynamically assigned to different RATs based on current service requirements and network capacity.
- Network operators and third party developers can create and sell their own custom RAT software applications, which can be implemented in any RRS.

Although RRSs are the subject of numerous research efforts, there is no unifying approach or set of requirements for an RRS architecture or framework [3, 5, 40]. Researchers and formal standardisation efforts each focus on specific topics relating to RRS, examples of which include how RRS-compliant hardware interfaces with RAT software and how RRS network elements within a RAN can be managed and controlled.

Although the ITU has expressed an interest in pursuing the development of an RRS standard, to date it has only published a very limited set of definitions for RRS related terms [41]. In contrast, the European Telecommunications Standards Institute

(ETSI) has taken a more proactive approach and published a set of technical reports on the topic of RRSs [7, 40, 42–44]; unfortunately these reports are limited to a set of recommendations for the future standardisation of RRSs, very basic early drafts of potential RRS architectures, and a functional architecture for the management of RRSs which was originally developed by research initiatives in Europe.

1.4 Thesis Objectives and Criteria

The purpose of this thesis is to design a new logical architecture using a greenfield approach, which includes and combines the functionality of all major existing RRSs, and which provides a unified RRS solution capable of autonomously managing and controlling the intelligent and secure dynamic reconfiguration of RATs within a RAN.

In addition, this architecture must be both flexible and scalable, and capable of implementing any legacy, current, and future RATs. The physical implementation of this architecture must also demonstrate that it is practically realisable and therefore capable of successfully transmitting and receiving a signal carrying data without errors.

As this target architecture contains the combined functionality of all major existing RRSs, it is referred to as the Converged Reconfigurable Radio System Architecture, or CRRSA.

It is important to note that the objective of the CRRSA is not to simply combine the functionality of all major existing RRSs, but rather to develop a completely new RRS architecture using a greenfield approach in order to avoid the limitations of existing RRSs. This approach is necessary as none of the existing RRS completely define and implement all RRS related functionality, and it would be impractical, and in many cases impossible, to simply modify or upgrade existing RRSs to provide a unified RRS solution.

In order to ensure that good engineering design practices are followed, the CRRSA must also conform to the following criteria:

- The CRRSA must be defined in an abstract manner, and not be tied to any specific technologies, albeit software or hardware.

- The CRRSA's components must be logically defined and distributed.
- The CRRSA's interfaces between its logical components must be well defined, thereby ensuring interoperability between, and portability of, all logical components and their functionality.
- The CRRSA must include a logical abstraction between each of the network domains (subscriber, access network, and core) and types of network traffic (user, signalling, and RRS management), thereby differentiating the functions performed within each domain and traffic supported by each logical component.

1.5 Constraints and Scope

As the objectives and criteria for the CRRSA are fairly broad, it is necessary to define the scope of the thesis, as well as the constraints placed on the modelling and implementation of the CRRSA. These are presented as follows:

- The CRRSA is limited to the air-interface in the subscriber and access network domains, and that portion of the core network that interfaces with the access network. Additional functions within the core network are abstracted. As most modern mobile communication systems are packet-based, the core network is assumed to be packet switched.
- For practical purposes, the CRRSA is implemented in a simulation composed of a combination of software and basic software defined radio hardware, and not in a real mobile communications network.
- The implementation of the CRRSA focuses on proving and demonstrating the architecture's functionality, rather than optimising the implementation for performance purposes.
- The modelling and implementation of RATs to demonstrate the CRRSA's capabilities is limited to a representative sample of two RAT models which incorporate the same digital signal processing techniques and wireless link control mechanisms specified by the IEEE 802.16 and 3GPP LTE RAT standards.
- The modelling of RATs is limited to the functionality needed to prove the concept is viable. As such, advanced communication technologies such as

spatial multiplexing, or dynamic link control, radio resource, and spectrum allocation algorithms, are excluded.

- All network traffic within the implementation, is modelled using random packets.
- The implementation of the CRRSA is limited to a link level simulation implemented on a PC with two Software Defined Radio (SDR) hardware platforms. Vertical and Horizontal Mobility, whilst supported by the architecture, is not implemented.

1.6 Thesis Outline

The remainder of this thesis contains five chapters, each of which covers the following topics:

- **Chapter 2**: examines the terms, technologies, research, and standardisation efforts relating to RRSs, and using the presented information, establishes a set of requirements for the CRRSA.
- **Chapter 3**: presents the CRRSA, describing the logical architecture, its components, and its interfaces.
- **Chapter 4**: discusses the 3GPP's LTE and IEEE's 802.16e RATs' air-interface specifications, and presents an abstracted model of each of the RATs.
- **Chapter 5**: implements the CRRSA in a practical and realistic environment, thereby demonstrating the CRRSA's management and control functionality, and its ability to support multiple RATs.
- **Chapter 6**: reviews and summarises the work covered in previous chapters, presents the key contributions made by this thesis, and discusses future work that could be performed to expand the scope and functionality of the CRRSA.

Chapter 2

Reconfigurable Radio Systems

Software Defined Radio (SDR) is relatively new technology that abstracts a RAT's functionality from its underlying hardware by implementing such functionality in software. Using SDR technology, a network operator is able to deploy a common hardware platform that can be reused across many different RATs, simply by changing the platform's software. (See [2–4, 6, 45–47])

Cognitive Radios evolved the SDR concept by adding additional functionality to enable the SDR to make intelligent decisions regarding its radio resource and spectrum utilisation [2,4,48,49]. Reconfigurable Radio Systems (RRS) take this concept a step further by providing network operators with the ability to incorporate, control and manage Software Defined and Cognitive Radios within a heterogeneous RAN [2,4,5].

The purpose of this chapter is to examine each of the major RRS research and standards efforts currently being undertaken, along with their supporting technologies. Thereafter, the objectives, design principles, and functionality provided by these RRSs and supporting technologies are compared, to determine a set of requirements for a new RRS. This new RRS, termed the Converged RRS, possesses a unique combination of RRS management and control capabilities, and a flexible software framework for developing reconfigurable radio applications and protocol stacks.

The remainder of the chapter proceeds as follows: Section 2.1 provides a background and set of concrete definitions for the SDR, Cognitive Radio, and RRS concepts, Section 2.2 examines each of the major RRS research and standards efforts, and discusses the concept of reconfigurable protocol stacks and provides an overview of the research currently being conducted in this field, and Section 2.3 critically analyses the work presented in previous sections and presented a set of requirements for the Converged RRS.

2.1 Software Defined Radios, Cognitive Radios, and Reconfigurable Radio Systems

The concept of a Software Defined Radio was first introduced by the United States Military in the 1970s, however practical realisation of such systems only became possible during the 1990s with the emergence of sufficiently powerful digital signal processing technologies [6, 50]. Cognitive Radios and Reconfigurable Radio Systems are far more recent concepts, having only been introduced within the last 10 years.

Many researchers make use of these terms interchangeably and concrete definitions are rarely provided. The following subsections briefly examine the ideas and capabilities behind each of these concepts, and provide a set of definitions which are used throughout the remainder of this thesis.

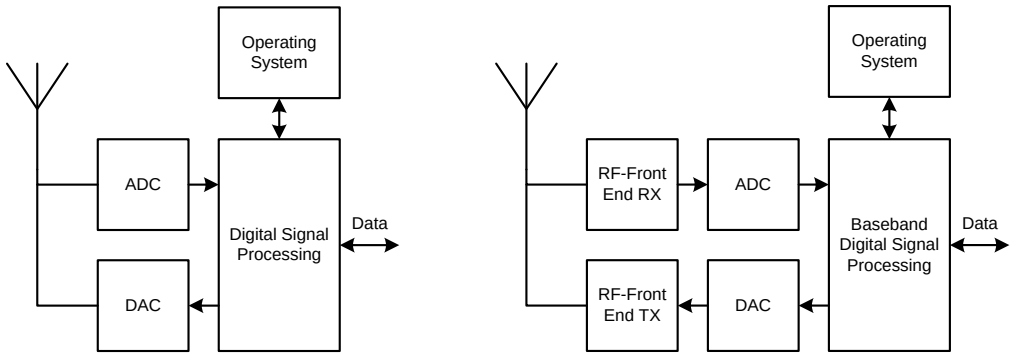
2.1.1 Software Defined Radios

In 1992 Joseph Mitola coined the term "Software-Defined Radio", which he described as: "[a radio that can] reconfigure itself to the appropriate signal format ... by running a different algorithm" [51]. An SDR differs from traditional radios, referred to as hardware-defined, in that various Radio Access Technologies (RAT) can be executed, either individually or simultaneously, on the same hardware platform simply by making changes to the system's software [4].

The first practical SDR systems were created by the US Department of Defence in 1992 under the Speakeasy I and Speakeasy II programs. These programs eventually evolved into the Joint Tactical Radio System (JTRS) program which is responsible for the development of the most widely adapted software architecture for SDR: the Software Communications Architecture (SCA) [52]. The SCA is examined in greater detail in Section 2.2.2. (See [3, 6, 50])

In 2009 the IEEE Standards Coordinating Committee 41 (SCC41) on Dynamic Spectrum Access (DSA) Networks, published a set of standards relating to topics involving emerging wireless networks, system functionality, and spectrum management. The IEEE 1900.1 [4] specification is one of these standards which provides a set of comprehensive definitions for SDRs, Cognitive Radios, and RRSs.

According to IEEE 1900.1, an SDR is defined as a "Radio in which some or all of the physical layers are software defined". An analysis of this definition notes that SDR functionality is limited to implementing a RAT's physical layer, and that such functionality is implemented in software. The fact that SDRs are only capable of implementing the physical layer is of particular importance, as it is the primary differentiator when comparing SDRs to Cognitive Radios and RRSs. In addition, it is also important to note that in order for a radio system to be classified as "software-defined", it must be possible to change the system's functionality post deployment, either by directly upgrading the software, or by downloading new software to the device or system [4].



(a) Ideal

(b) Realistic

Figure 2.1: SDR Architectures (Adapted from diagrams and information in [2, 4–6, 47, 49])

To further understand how an SDR operates, it is necessary to briefly examine the components that are commonly used in most SDR systems. Figure 2.1a illustrates the ideal SDR system architecture. The components in this architecture are briefly described as follows [2, 42, 46, 47, 49]:

- **Antenna or Antenna Array**: Since an SDR must be capable of producing and processing any waveform at any frequency, broadband antennas capable of receiving and transmitting signals over a wide range of frequencies are required.
- **ADC and DAC**: An Analogue-to-Digital Converter (ADC) converts the digital signal into an analogue form for transmission, and a Digital-to-Analogue Converter (DAC) converts the received analogue signal into a digital form for processing.

- **Digital Signal Processing**: A combination of digital signal processing software and hardware is responsible for implementing each RAT's physical layer. Hardware components could include Field Programmable Gate Arrays (FPGA), Digital Signal Processors (DSP), and System on a Chip (SoC). The software is usually implementation specific, and is commonly referred to as an SDR application, waveform, or waveform application.
- **Operating System**: General management and control functionality is provided by the SDR's operating system, and is usually executed on separate hardware from that used for digital signal processing; an example of which could be a General Purpose Processor (GPP).

The ideal SDR architecture is simplistic and benefits from enabling the transmission frequency, gain, and digital transmission filters to be defined and configured in software; however this places exceptionally high performance demands on the ADC and DAC components. Such extreme requirements result in very costly SDR implementations, a fact which has encouraged researchers to develop more practical architectures, as illustrated in Figure 2.1b. (See [2, 6, 47])

In a realistic SDR system architecture, an RF front-end performs the conversion between the analogue Radio Frequency (RF) signal transmitted and received by the SDR, and the analogue baseband signal that can be processed by the SDR's other components. Although RF front-ends can include software tuneable components, they do not afford the same flexibility as an ideal SDR system. (See [2, 6, 47])

Practical SDR systems also provide a software framework for developers to create, deploy and manage radio applications. The exact nature of this framework varies greatly between implementations; however it usually takes the form of an Application Programming Interface (API) which provides the means to access and control the SDR's digital signal processing elements, and it exploits the services provided by the SDR's operating system. Examples of such APIs include the Common Object Request Broker Architecture (CORBA) interface as used by the SCA, discussed in Section 2.2.2, and the Universal Hardware Driver (UHD) for the Universal Serial Radio Peripheral (USRP), discussed in Chapter 5.

Despite the fact that SDR research has made significant advances since the first systems appeared in the early 1990's, there are still significant challenges that are limiting the widespread adoption of SDR technology [2, 3]:

- **Standardisation and interoperability**: Although there are several widely adopted standards for SDR, there is no universally accepted standard. This results in SDR software and hardware that is unique to each standard, and is largely incompatible or not portable.

- **Computational requirements**: The digital signal processing requirements for SDR make it extremely difficult to develop practical systems with reasonable power consumption, cost, and physical size.

- **Security**: Unlike hardware-defined radios, SDRs possess all of the vulnerabilities associated with conventional personal computing, such as the risk of hacking and the installation of malicious software. These challenges increase when a SDR possesses the capability to download and update its software over the air (OTA).

- **Certification [Regulatory]**: National telecommunications regulatory bodies must certify RAT equipment before it can be deployed in each country. The purpose of such certification is to ensure that the RAT meets particular performance requirements, and to ensure that different RATs do not interfere with one another. In contrast to traditional radio systems however, an SDR is not limited to the implementation of a particular RAT but can rather implement multiple RATs, which creates significant challenges for both regulators and vendors when attempting to certify SDR systems using existing certification processes.

While it will most likely still take several years until SDRs are widely deployed, the advantages of SDR technology are universally acknowledged. Some of its key advantages are [2, 3, 5, 6, 45]:

- **Common Hardware Platforms**: As each new RAT is developed, standardised, and deployed, network operators and subscribers are currently required to invest in new telecommunications equipment to support implementation of the RAT. SDR technology offers the opportunity to avoid such costly expenditure by enabling the same hardware to be reused for different RATs. The use of a common hardware platform also results in lower development costs, lower manufacturing costs due to greater mass production, and increased optimisation of the hardware as it is constantly reused.

- **Standardised Software**: When developing new RATs, the time taken to implement such RATs is greatly reduced, as developers can make use of pre-existing software written for previous RATs. The reuse of software also results in the optimisation of common software components which can improve the SDR system's performance.

- **Scalable**: A SDR system permits numerous RATs to be deployed on the same hardware, thereby simplifying integration. Software upgrades can be used to install new RATs as they become available; making the system future-proof.

- **Remote Software Deployment**: The use of software to define RAT functionality means that network operators and subscribers can upgrade their equipment simply by downloading a software update. This functionality extends to the possibility of downloading such software OTA using an existing operational RAT. It should be noted however, that OTA software downloads and installations require a network operator to provide a secure software distribution environment, which would usually form part of a RRS.

- **Support for CR and RRSs**: SDRs are a key enabling technology for both Cognitive Radios (See Section 2.1.2) and RRSs (See Section 2.1.3), as they provide the means whereby such systems can adapt their physical layer functionality in order to implement various RATs.

2.1.2 Cognitive Radios

In 2000 Joseph Mitola, whilst writing his PhD Thesis, first introduced the concept of a Cognitive Radio [53]. A Cognitive Radio is a radio which possesses the ability to collect information regarding the wireless environment and the internal state of a network element, and utilise such information to determine how the network element should adapt its current functionality in order to deliver a certain level of Quality of Service (QoS) or to achieve a set of predefined objectives. (See [2, 4, 48, 54])

The operations performed by a Cognitive Radio can be broken down into three steps, which are implemented by the logical components presented in Figure 2.2 [2, 6, 54]:

- **Context Awareness**: A Cognitive Radio collects contextual information relating to the quality of the wireless environment, spectrum availability,

achieved quality of service, user preferences, cost of connecting to each available network operator, and any other relevant information relating to the radio's environment and its own internal performance.

- **Decision Making**: Using the context information collected, a Cognitive Radio determines whether the radio's functionality should be adjusted to meet predefined objectives or to maintain specific QoS levels. Such objectives could take the form of specific rules or network operator-defined policies.

- **Adjusting Radio Functionality**: A Cognitive Radio adjusts its functionality to implement decisions made by effecting changes to the Physical and Data Link Layers in the protocol stack. It is possible to also modify the functionality contained within higher layers, provided that they support reconfiguration operations.

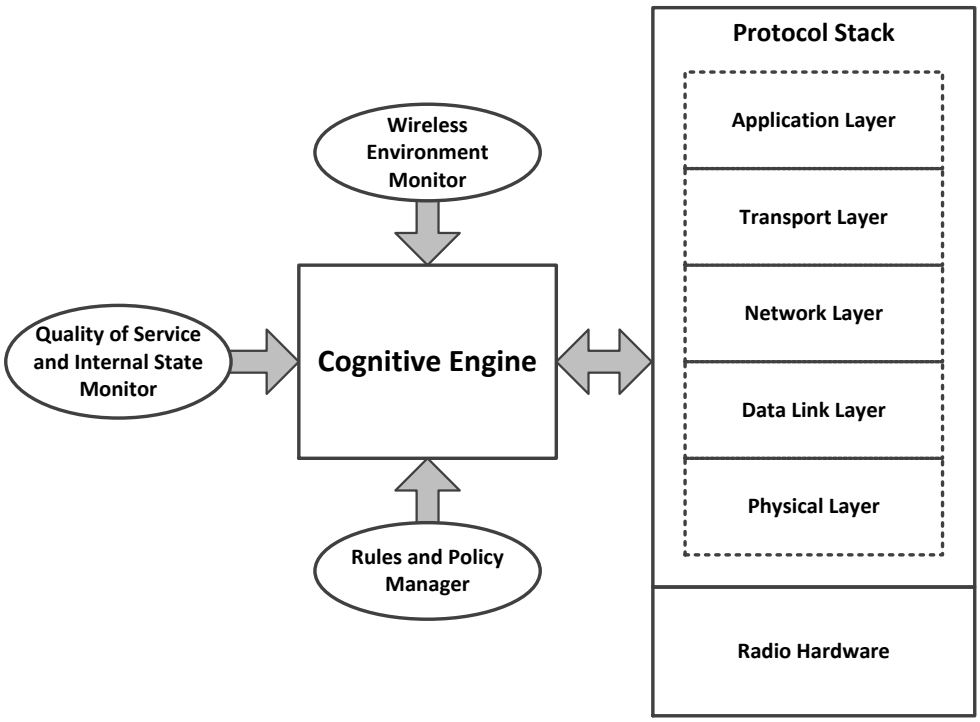


Figure 2.2: The Cognitive Radio concept [2, 6, 48, 54]

The manner in which these operations are performed varies greatly between different Cognitive Radio implementations as there is no universally accepted standard for such radios. For example, some researchers have indicated their belief that a Cognitive Radio could be implemented without the use of SDR technology [4,48]. In

such a case, the Cognitive Radio would still be able to collect context information and make decisions to adjust functionality as required, however the degree to which such a system would be able to implement such adjustments would be limited to simple parameter and configuration changes. Other researchers have proposed Cognitive Radio systems which combine the functionality of an SDR platform with the ability to continuously learn and improve the system's decision making capabilities [2,48,54].

Researchers agree that Cognitive Radio implementations must be capable of performing their operations in real-time, otherwise any adjustments made may no longer be relevant due to rapidly changing environmental conditions [2, 4, 6, 54].

Cognitive Radios have also been proposed as the key enabler for a new approach to managing access to the radio spectrum known as Dynamic Spectrum Access (DSA). As the number of wireless devices and technologies increases, the amount of available radio spectrum decreases. Although the majority of the radio spectrum has already been allocated by international and national regulatory bodies, only a small portion of this spectrum is actually utilised at any specific place and time.

DSA aims to address and exploit this issue by using Cognitive Radio technology to enable radios to dynamically adjust which portion of the spectrum they use. This opens up a number of potential scenarios, including [2, 5, 48, 49, 54]:

- **Dynamic Spectrum Allocation**: RANs can dynamically adjust how much of the spectrum they occupy based on demand.
- **Spectrum Leasing**: RANs can offer to share or rent the available spectrum with other network operators' RANs.
- **Spectrum Sharing**: RANs can allow unaffiliated Cognitive Radios to opportunistically make use of licensed spectrum when a licensee is not presently utilising it.

Cognitive Radios currently suffer from all of the same challenges faced by SDRs, including a lack of standardisation, high computation requirements, numerous security threats, and problems in obtaining the required certification [2, 3, 6, 54].

In the past years, there has been a substantial increase in research in the field of Cognitive Radio systems, as their decision making capabilities can be coupled with SDR systems to create extremely flexible radio systems that are able to autonomously adapt their functionality based on current conditions and a set of

predefined objectives. To fully exploit these capabilities however, Cognitive Radios and SDRs cannot operate in isolation and must be combined with a management and control system to coordinate the autonomous reconfiguration of radio elements and their resources within a RAN. Such a system is referred to as a Reconfigurable Radio System (RRS), and is examined in the following subsection.

2.1.3 Reconfigurable Radio Systems

Reconfigurable Radio Systems (RRS), which are also referred to in some literature as Cognitive or Reconfigurable Networks, are an evolution of the SDR and Cognitive Radio concepts [2,4]. An RRS provides the functionality required to coordinate and control the operations of multiple Software Defined and Cognitive Radio network elements, giving the network operator the means to jointly manage and optimise radio resources and spectrum utilisation within a RAN [2, 55]. Reconfigurable network elements within an RRS can be located in either reconfigurable mobile devices (RMD) or reconfigurable base stations (RBS) which contain SDR and Cognitive Radio capabilities.

When managing the reconfiguration operations performed within a RAN, an RRS must perform similar functions to those of Cognitive Radios; however an RRS's operations take place on a much wider scale and are more complex due to the numerous reconfigurable network elements. These operations can be summarised as follows [2, 5, 43, 55–57]:

- **Network-wide Context Awareness**: An RRS must gather context information from each reconfigurable network element relating to the quality of the wireless environment, the spectrum available for, and the spectrum allocated to each RAT, the network capacity within each cell, the requirements for services provided to subscribers, the RATs currently deployed, and the network elements that are capable of supporting each RAT.
- **Negotiated and Regulated Decision Making**: RRSs must provide the infrastructure for network elements to be able to negotiate amongst themselves, so as to determine whether any specific network element is allowed to adjust its functionality, and based on this decision either instantiate or remove a RAT. These negotiations must be regulated according to rules and objectives specified by network operators, and must comply with spectrum access regulations. In this way, network operators are able to specify rules

and parameters that are used to autonomously manage and optimise their radio resources and spectrum utilisation, whilst still permitting each network element the freedom to adapt its functionality based on its own requirements.

- **Reconfiguration**: Each network element must implement the decision reached following negotiations performed via the RRS, by adjusting its parameters and its current RAT.

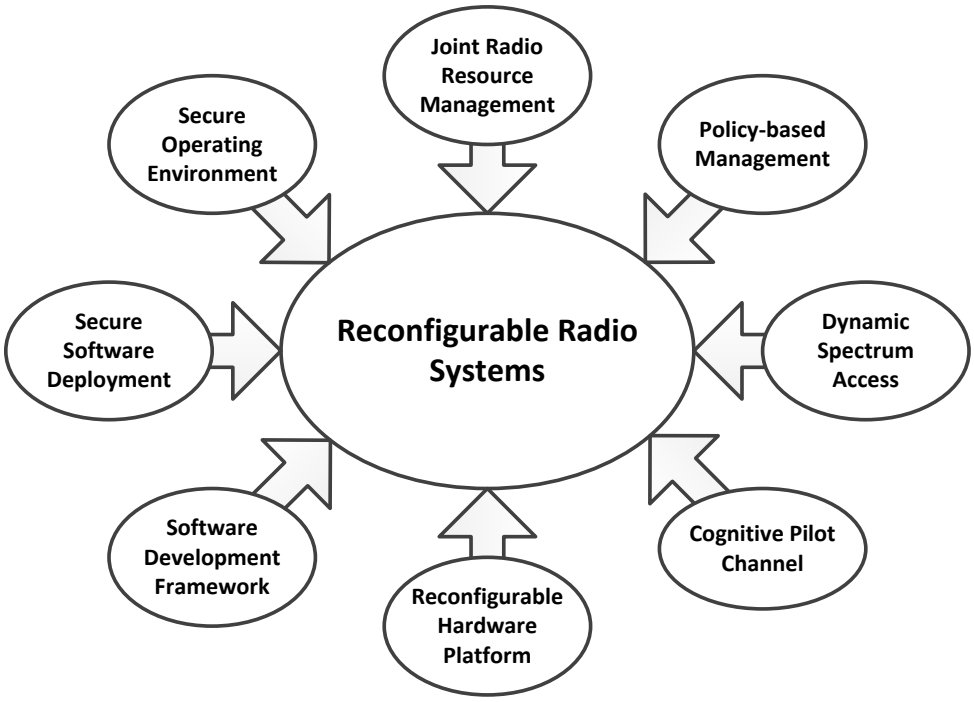


Figure 2.3: The logical elements of Reconfigurable Radio Systems [2, 5, 43, 55–59]

Figure 2.3 presents the various elements of a RRS which are needed in order to be able to perform its reconfiguration management operations. The joint radio resource management element is responsible for coordinating the negotiations between reconfigurable network elements, and for ensuring that the optimal utilisation of radio resources by all network elements is achieved. (See [2, 5, 43, 55–59])

The RATs in each network element possess their own individual resource management, mobility management, and other link control mechanisms, which are in turn defined within their RAT standards. The RRS's reconfiguration management system interfaces with these mechanisms, thereby enabling it to adjust the radio resource utilisation of each network element.

A network operator is able to exercise control over reconfiguration operations through the use of policies, which provide network elements with a set of predefined reconfiguration objectives and restrictions. Policy-based management is also beneficial, as it enables the autonomous reconfiguration of network elements without needing to request / obtain permission from the network operator in the core network. (See [2, 5, 43, 55–59])

DSA enables the dynamic allocation of spectrum to accommodate each reconfigurable network element, and in so doing ensure both efficient use of the available spectrum and adherence to national spectrum licensing regulations. DSA elements in separate RRSs can also conduct negotiations, thereby enabling spectrum leasing and sharing. (See [2, 5, 48, 49, 54])

Due to the reconfigurable nature of RMDs and RBSs, establishing communications between these network elements is impossible if the RMD is unaware of the RAT currently implemented by the RBS [6,55]. To resolve this problem, RRS researchers have developed the concept known as a Cognitive Pilot Channel (CPC). A CPC is a universal predefined logical communications channel, which enables a RMD to contact a RBS and determine which RATs may be implemented by the RMD in order to establish communications with the RBS [44].

Apart from facilitating initial communications between RMDs and RBSs, the CPC can also be used to convey all reconfiguration management messages within an RRS, or support negotiations for spectrum sharing from secondary users of licensed radio spectrum [6, 55]. The CPC can be established as a logical channel over an existing RAT (in-band), or via a new RAT dedicated to performing this function (out-of-band) [44]. The CPC is discussed in more detail within an ETSI technical report which was written as a prelude to the development of a CPC standard [44].

To ensure interoperability between network elements and portability of software, an RRS must define a standardised interface that connects the reconfigurable hardware platform to the operating system and radio applications within each network element. This principle extends to the requirement for a standardised software development framework for radio applications. Such a framework enables radio application developers to create RAT software which can be deployed on any network element provided that such elements contain hardware capable of meeting the software's performance requirements. (See [2, 5, 6, 43, 55–59])

Unlike traditional mobile communication systems, where the functionality is static and the configuration of each device's parameters is limited, the reconfigurable nature of network elements within an RRS introduces numerous security vulnerabilities. Examples of potential security threats are [2, 3, 5, 58]:

- Malicious software can be downloaded and installed on a network element. Such software could completely prevent the device from operating, steal confidential information, adjust the functionality of the network element such that it interferes with the operation of other mobile communication devices, or enable a malicious individual to obtain control over the network element and its operations.
- Deliberate tampering or errors introduced during the downloading of new policies can result in the network element being incorrectly reconfigured. This could result in decreased performance by network elements, RMDs being unable to communicate with RBSs, or network elements interfering with mobile communication devices.
- Radio applications and other proprietary software could be downloaded from a network element thereby facilitating intellectual property theft.

To combat the security threats introduced by RRSs, secure methods for deploying and installing software in network elements, and a secure operating environment within each network element, are both required. These methods include the use of encryption and authentication when downloading and installing new radio applications, certification of radio applications prior to their installation on a network element, encryption of personal information on a RMD, and secure immutable logical components within a network element that can detect security threats and prevent the device from operating until the threat has been neutralised. (See [2, 3, 5, 58])

Currently there are numerous RRS research and standardisation initiatives, each of which has developed its own frameworks, architectures, and standards to meet its own requirements. The following section examine these initiatives, and focus on the functionality they provide.

2.2 Reconfigurable and Software Defined Radio Research and Standardisation

The purpose of this section is to introduce and then examine the functionality provided by all of the major RRS research and standardisation initiatives. The focus of each of the following subsections is on presenting the requirements, architecture, and general functionality provided by the logical components within each RRS. An in depth study of the methods and technologies used to model and implement each RRS is considered outside of the scope of this thesis, as such concepts are specific to each of the described RRS and are unnecessary for determining the requirements for, and developing, the CRRSA.

The research and standardisation initiatives discussed in the following subsections can be broadly classified into four categories:

- **SDR Architectures**: focus on the interfaces and logical components needed to implement individual SDRs and radio applications. ETSI has published some early drafts of possible SDR architectures, which are presented in Section 2.2.1, however the Software Communications Architecture is the most widely adopted SDR specification and this is presented in 2.2.2.
- **Management and Control of RRSs**: which includes all of the logical elements needed to manage and control all reconfigurable elements within a RAN. The two primary research and standardisation initiatives in this category are the Functional Architecture for the Management and Control of RRSs, presented in Section 2.2.3, and the IEEE's 1900.4 standard, presented in Section 2.2.4.
- **Commercial Base Station Architectures**: whose primary purpose is to split each base station into a system unit and a remote radio head. The two major industry standards for commercial RRSs are the Open Base Station Architecture Initiative (OBSAI) and the Common Public Radio Interface (CPRI), both of which are discussed in Section 2.2.5.
- **Reconfigurable Protocol Stacks**: focus on decomposing a RAT into individual functions, which can be reused and reassembled to implement flexible radio applications. Although there is no major research or standards initiative actively engaged in this field, there are a number of independent researchers

who have conducted their own investigations into the subject. A summary of these investigations is presented in Section 2.2.6.

2.2.1 The ETSI Technical Committee for RRSs

In 2008 ETSI established a Technical Committee for RRSs. The purpose of the committee is to investigate SDR and Cognitive Radio technologies as components of RRSs, and to develop standardised approaches and architectures for the deployment of RRSs [60].

The committee is divided into four working groups, each of which focuses on a particular aspect of RRS standardisation [61]. Since its creation, the committee has only released a few technical reports (in 2009) which contained early draft specifications, and recommendations for future RRS standardisation initiatives by ETSI [7, 40, 42–44]. Each of the reports was compiled using information generated by other major RRS organisations and standards, many of which are examined in the remaining sections of this chapter.

Section 2.2.3 discusses the committee's specification for a Functional Architecture for the Management and Control of RRSs, as this forms part of standardisation efforts undertaken by the E3 European Research programme focusing on RRSs.

The following subsections provide an overview and examine the functionality of the ETSI SDR Reference Architecture and the ETSI Reconfigurable Architecture for Radio Base Stations.

2.2.1.1 ETSI SDR Reference Architecture

The ETSI SDR Reference Architecture is published in ETSI Technical Report 102.680 [42] and is based on a series of requirements determined through the examination of various existing SDR standards and research initiatives. These requirements are summarised as follows:

- **Architectural Requirements**: The architecture must cover all functionality from the antenna through to the network layer in the protocol stack. Development should make use of good software engineering practices, specifically model-driven and component-based development methodologies. This

approach ensures that any implementation of the architecture results in a modular, well-defined system that guarantees interoperability between components, regardless of the manufacturer, and enables portability of software between platforms.

- **Operational Requirements**: The architecture must make provision for a wide variety of Radio Access Technologies, and must possess the capability to dynamically reconfigure a standardised platform to implement the required RATs. From a physical layer perspective, the architecture must also support multiple frequency bands across the radio spectrum. A secure operating environment is also of paramount importance.
- **Capability Requirements**: The architecture should support the ability to install and execute radio applications whilst running an existing application, to execute multiple RATs simultaneously, and to allocate and manage radio resources between numerous RAT instances.

Based on these requirements, the ETSI Technical Committee on RRSs produced their SDR Reference Architecture for individual mobile devices, which is illustrated in Figure 2.4. The architecture contains a series of logical components, termed the Control Framework, which provides the following control and management functionality:

- **Configuration Manager**: loads/unloads radio applications and manages radio application parameters.
- **Radio Connection Manager**: activates and deactivates radio applications based on user requests and requirements.
- **Flow Controller**: send and receive user data and control flow.
- **Multi-Radio Controller**: schedules requests for spectrum resources from applications running in parallel.
- **Resource Manager**: manages radio resources between multiple applications in order to meet their requirements.
- **Administrator**: Allows admin users to install new radio applications.
- **Mobility Policy Manager**: Maintains the user preferences and policies regarding different RATs and makes the selection between them.

In addition to the Control Framework described above, the architecture also includes a number of software interfaces designed to ensure interoperability between the architecture's components, and portability of radio applications:

- **Multi-Radio Access Interface**: provides services required to interface user applications with the Control Framework. Such services include radio access control, to determine which RAT to deploy, data flow services which connect the network protocol stack and radio applications, and administrative services used to install and manage radio applications.
- **Unified Radio Application Interface**: standardises the interactions between radio applications and the remainder of the architecture. This interface includes services to manage radio applications and their configurations, send and receive data, and schedule and control radio resources between radio applications.
- **Radio Programming Interface**: provides a standardised set of interfaces used by developers to create new radio applications while ensuring platform independence and portability of such applications. This interface includes a common programming model and software libraries. Please note that this interface is not shown in Figure 2.4.
- **Reconfigurable Radio Frequency Interface**: is a potential interface that could be included in future standardisation initiatives by ETSI. The purpose of this interface is to provide radio applications with access to different Radio Frequency circuits depending on the requirements of each radio application.

A detailed description of the SDR Architecture, its components, and its interfaces has yet to be published by ETSI; however the Technical Committee on RRSs has indicated its intention to formally standardise the SDR Architecture at some stage in the future.

2.2.1.2 ETSI Reconfigurable Architecture for Radio Base Stations

The Reconfigurable Architecture for Radio Base Stations, as published in the ETSI Technical Report 102.681 [40], is still in the very early stages of development, and consists of a set of requirements divided into various categories, and a list of potential capabilities together with an abstracted architectural diagram. The requirements are as follows:

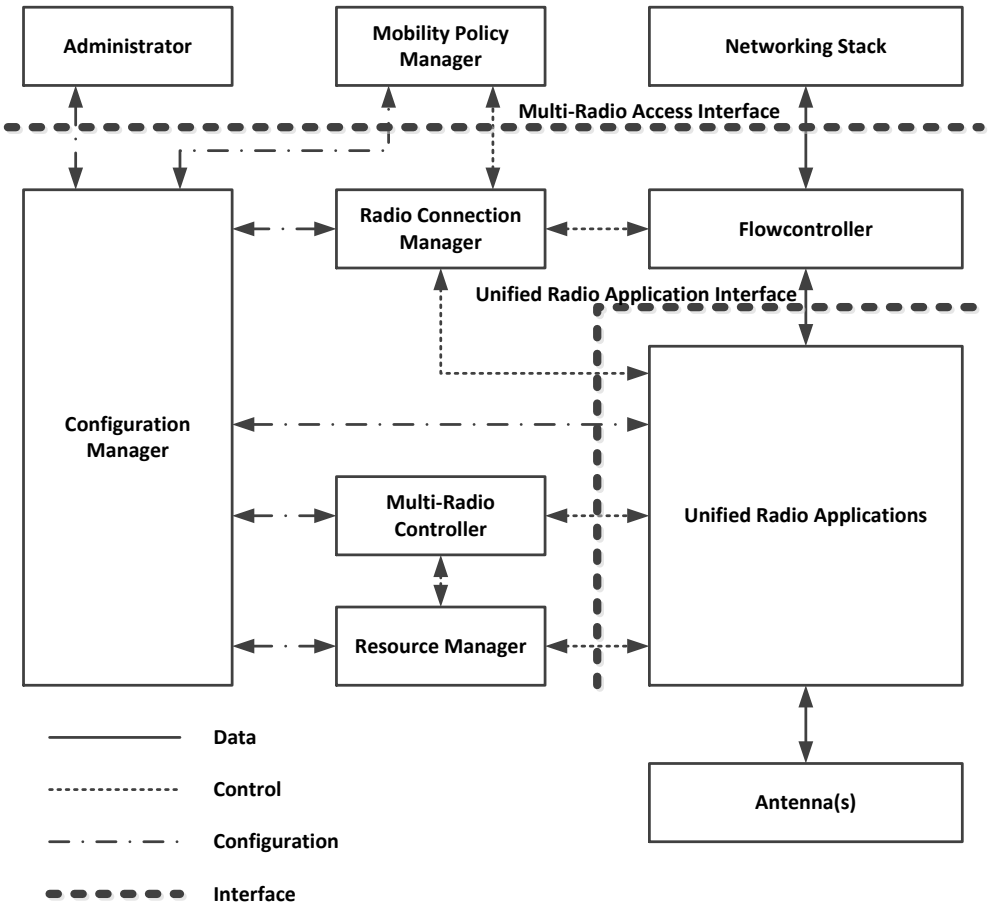


Figure 2.4: The ETSI SDR Reference Architecture [42]

- **Generic Requirements** include the functionality needed to perform reconfigurations in order to implement different RATs, support multiple bandwidths and frequency bands, dynamically adjust which spectrum is used for transmission, optimise capacity based on load and QoS requirements, perform network planning and optimisation, perform antenna tuning, and backhaul configuration.

- **Operator Requirements** include the functionality needed to: jointly manage the configuration and resource management of base stations which co-exist within the same geographical area, and the ability to support network planning deployment and management in an effort to reduce Capital Expenditure (CAPEX) and Operational Expenditure (OPEX).

- **OEM Requirements** include the functionality needed to: upgrade existing systems using software, increase capacity of systems via hardware upgrades, and a reliability and certification programme for all software.

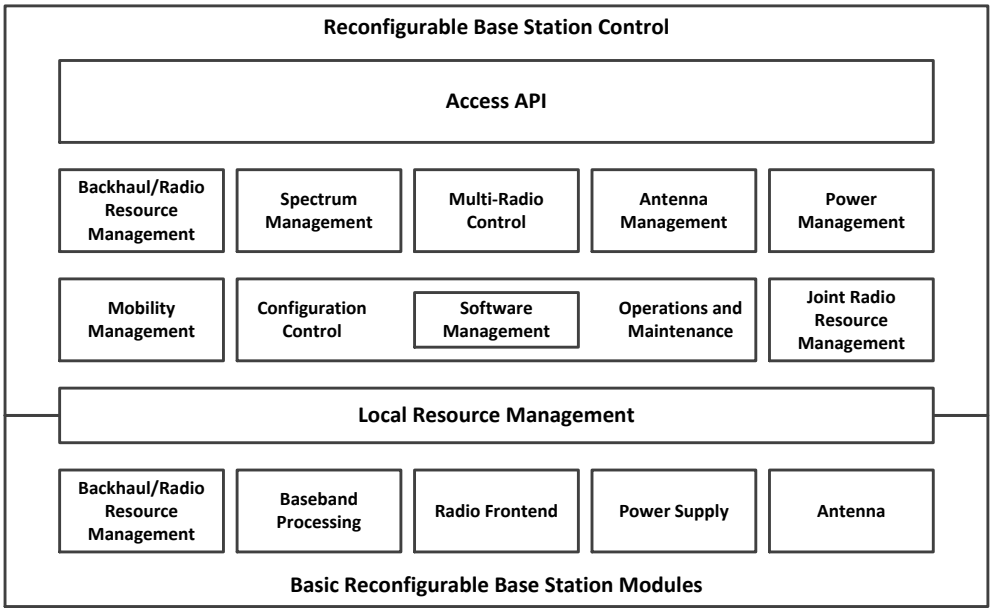


Figure 2.5: ETSI Reconfigurable Architecture for Radio Base Stations [40]

Taking into account these requirements, as well as information garnered from existing RRS standardisation and research initiatives, the ETSI Technical Committee on RRS determined that each Radio Base Station should provide the following capabilities, which are shown in an abstract architectural diagram in Figure 2.5:

- **Configuration Control**: triggers and manages the reconfiguration of the base station.
- **Software Management**: downloads and manages all software used by the base station. Manages the process used to construct radio applications in the event that a component-based approach is implemented.
- **Operations and Maintenance**: collects and manages all information regarding the base station's operations and current configuration.
- **Backhaul Management**: consists of the physical and logical components used to implement a specific backhaul solution, e.g. SDH, Ethernet, etc.

- **Spectrum Management**: assigns and manages spectrum based on available policies. Also provides functions needed to implement cognitive radio functionality.
- **Multi-Radio Control**: manages and controls multiple RATs in an efficient manner.
- **Antenna Management**: determines, implements, and manages the appropriate antenna configuration.
- **Power Management**: manages and controls the base station's transmission power. Provides RATs with an interface to adjust transmission power for mobility management functionality.
- **Mobility Management**: manages vertical (between RATs) and horizontal (between cells using the same RAT) handover procedures.
- **Radio Resource Management**: supports RAT selection and configuration based on QoS requirements, wireless environment conditions, RAN traffic conditions, network policies, and subscriber preferences.

In similar fashion to the ETSI SDR Architecture, the Technical Committee for RRSs has not provided any significant specifications for its Reconfigurable Architecture for Radio Base Stations.

2.2.2 The Software Communications Architecture

The Software Communications Architecture is a software framework designed to assist in the development of individual SDR systems. The SCA specification was first released in 2000 [50,62], and has subsequently undergone several revisions, with the most recent version having been released in May 2006 [52, 62].

The SCA was created by the Joint Program Executive Office (JPEO) of the Joint Tactical Radio System (JTRS) programme within the United States Department of Defence, in collaboration with various non-profit standards organisations, including the Wireless Innovation Forum and the Object Management Group [3, 49, 50].

Although the framework was originally developed for use by the military, its capacity to support flexible multi-band multi-mode SDRs has led to the SCA becoming the most widely adopted software framework for the development of SDR systems.

Implementations of the SCA can be found in both the academic and commercial sectors. Examples of such implementations include OSSIE (Open-Source SCA Implementation::Embedded) from Virginia Tech [63], and the Spectra product line from PrismTech [64].

Widespread deployment of the SCA within the commercial sector has however been slowed by the specification's prioritisation of military objectives, such as flexibility, interoperability and portability over the requirements of more commercially-oriented organisations, which are more interested in prioritising factors such as performance, cost, and energy efficiency [63].

2.2.2.1 Overview of the Software Communications Architecture

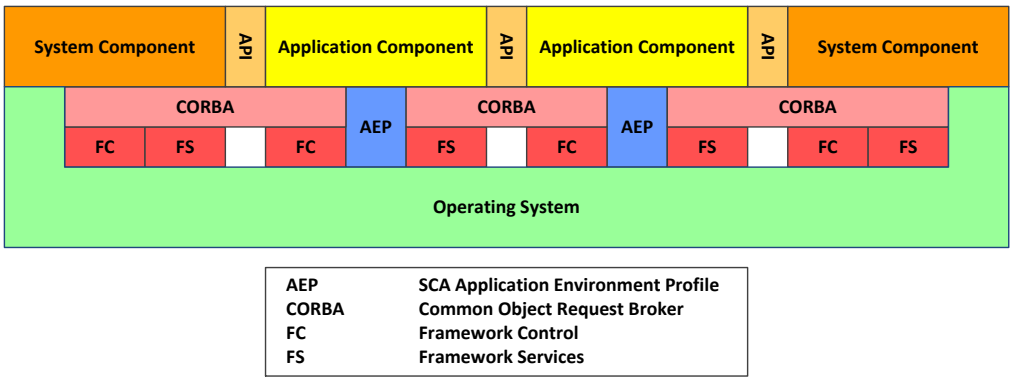


Figure 2.6: The Software Communications Architecture [3, 50, 52]

The SCA architecture, seen in Figure 2.6, provides an operating environment which consists of an operating system, middleware, and the interfaces used to implement and manage SDR applications, collectively known as the Core Framework. (See [3, 49, 50, 52, 63])

SDR applications implemented using the SCA are referred to as "Waveform Applications", or simply "Waveforms". Each waveform is constructed from a library of software components, each of which is written based on the SCA specification. This methodology enables SDR application developers to reuse components when creating new waveforms, thereby decreasing overall development time and cost. (See [3, 49, 50, 52, 63])

The SCA's operating system is derived from the Portable Operating System Interface (POSIX) specification developed by the IEEE . Waveforms can gain access to a limited subset of POSIX functions through the Application Environment Profile (AEP). (See [50, 52, 63])

The purpose of the SCA is to create a software framework which allows developers to create SDR systems that possess the following characteristics [3, 46, 63]:

- **Abstraction from the underlying hardware**: SDR application developers using the SCA framework do not have to concern themselves with hardware related issues. Conversely, SDR platform developers do not have to concern themselves with SDR application matters.
- **Hardware agnostic**: the SCA does not specify the hardware components or architecture that are used in any implementation. The SCA is capable of adapting itself to the hardware used for each implementation, even those which are distributed in nature.
- **Portable SDR applications**: SDR applications written according to the SCA framework can be executed on any SDR platform. It should be noted however that SDR platforms must be capable of meeting each SDR application's processing requirements.
- **Reuse of software components**: the SCA makes it possible to reuse components of existing SDR applications when developing new applications. This reduces overall development time and enables SDR application developers to reuse software components from different sources.
- **Capable of implementing any RAT**: the SCA does not place any restrictions on SDR applications. This means that it is theoretically possible to implement any Radio Access Technology (RAT) using the SCA.
- **Compatible with commercial off-the-shelf (COTS) systems**: the SCA employs widely adopted software engineering technologies and principles, thereby promoting the use of COTS systems and reducing the cost of SCA implementations.

2.2.2.2 The Software Communications Architecture's Middleware

The OMG's Common Object Request Broker Architecture (CORBA) was chosen by the JTRS programme to be the SCA's middleware. CORBA is an open standard published by the OMG, and its purpose is to enable software components written in a variety of programming languages, and potentially distributed across numerous processors or separate networks, to operate together as if they were part of a single application [46, 62].

CORBA's role within the SCA is to administer communications between the framework's distributed software components, and provide an abstraction between waveform applications and the underlying hardware. The benefits of CORBA are that waveform application developers can make use of any supported programming language when creating new software components, and SDR platform developers are not restricted to specific hardware components or configurations, thereby satisfying the criteria that the SCA be hardware agnostic [3, 50, 52, 62, 63].

The interface between the CORBA middleware layer and specialised hardware components such as DSPs and FPGAs is not standardised by the SCA, as the vast majority of such components do not support CORBA [3, 46]. SCA compliant SDR platform developers are therefore required to develop their own proprietary interfaces, which can affect SCA software portability and hardware independence. In an effort to resolve the situation the JTRS created the Modem Hardware Abstraction Layer (MHAL) specification which provides a low-level interface for specialised hardware, however it has been noted that the MHAL is not as robust as CORBA [46].

Although CORBA is an integral part of the SCA as it fulfils the framework's requirements for an open, flexible and scalable software architecture, the use of CORBA has resulted in strong criticism of the SCA as many researchers believe that it significantly increases processing demands and therefore decreases the performance of any SCA implementation [3, 63]. Proponents of the SCA have rejected such claims, stating that although CORBA does increase each implementation's processing requirement, the performance reduction can be mitigated through the careful choice of a basic transport mechanism to be used by CORBA, rather than the default TCP/IP based transport mechanism [3, 50]. Despite the controversy over the choice of CORBA, it is important to note that at present there are no alternative technologies which are capable of satisfying the SCA's middleware requirements [62].

2.2.2.3 The Software Communications Architecture's Core Framework

The SCA's Core Framework (CF) is composed of a set of interfaces, defined using CORBA's Interface Description Language (IDL), which are used to manage and control waveform applications, and abstract them from the underlying hardware. The four types of interfaces contained within the CF are as follows:

- **Base Application Interfaces**: are implemented by software components used to construct waveforms, and provide management and control functionality.
- **Base Device Interfaces**: abstract hardware from the rest of the system, by providing a logical representation of hardware components which can be used to manage and control such hardware.
- **Framework Control Interfaces**: are responsible for managing all software components within the framework, by providing the functionality required to instantiate, configure, and destroy software components.
- **Framework Services Interfaces**: provide additional support functionality to the framework.

Figure 2.7 illustrates, and indicates the relationship between, all of the Base Application Interfaces and some of the relevant Framework Control Interfaces. The Resource interface provides common functionality needed to manage and control software components. The Resource interface's ability to initialise and release, perform built-in tests of its functionality, allow the configuration and querying of its properties, and provide a reference to one of its Port interfaces, is achieved through its inheritance from the *LifeCycle*, *TestableObject*, *PropertySet*, and *PortSupplier* interfaces respectively. (See [50, 52, 63])

The Port interface allows different Resources to communicate with one another, and is the mechanism by which software components are combined to form waveform applications. Port interfaces are divided into different categories depending on the type of traffic they transport, and on whether they act as a client or server during inter-component communication. Waveform applications are represented by the Application interface, which acts as a container for all the Resources used to create the application. (See [50, 52, 63])

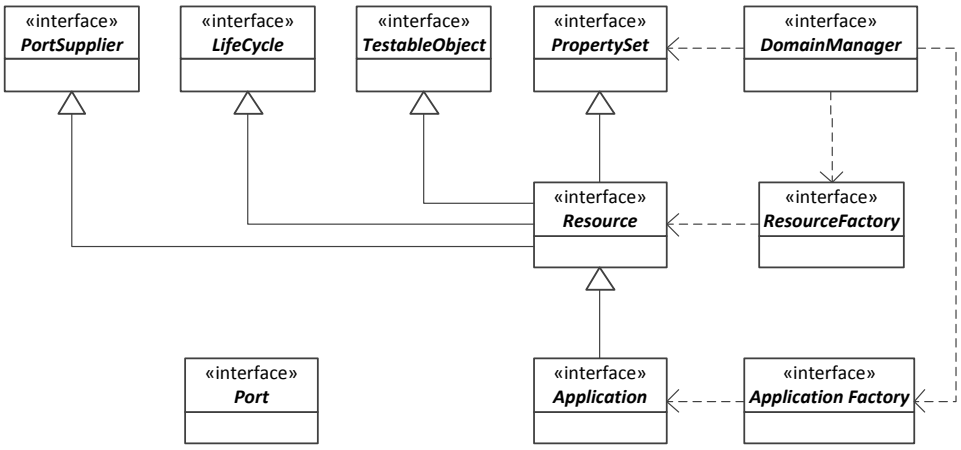


Figure 2.7: The Software Communication Architecture's Core Framework Base Application and Framework Control Interfaces [3, 50, 52, 63]

The *ResourceFactory* and *ApplicationFactory* are Framework Control Interfaces used to create and destroy Resources and Applications respectively. The *Application-Factory* is also responsible for obtaining a list of the required Resources from the *DomainManager*, and generating the connections between them. (See [50, 52, 63])

The *DomainManager*, also a Framework Control Interface, provides the SCA framework with all the necessary control and management functionalities for the entire system. The *DomainManager* initialises and launches the primary CORBA services, coordinates with the other Framework Control Interfaces during the SCA's operation, and creates *FileManagers* which contain the instructions needed to manage and control the system's resources. (See [50, 52, 63])

Figure 2.8 illustrates all of the Base Device Interfaces and some of the relevant Framework Control Interfaces, and indicates the relationship between these interfaces. The Device interface provides a logical representation of a hardware component. It inherits from the Resource interface, and incorporates the additional functionality needed to monitor and manage hardware components. The Device-Manager Framework Control Interface manages and controls all Device interfaces. (See [50, 52])

The *LoadableDevice*, *ExecutableDevice*, and *AggregateDevice* interfaces each extend the Device interface's functionality by including functions needed to monitor and control different types of hardware components such DSPs, FPGAs, GPPs, and composite devices. (See [50, 52, 63])

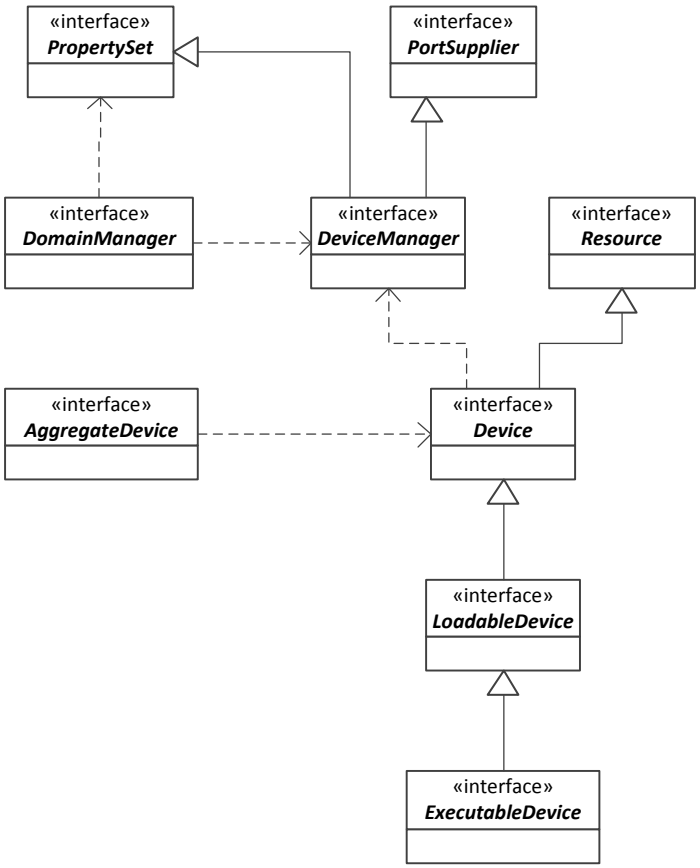


Figure 2.8: The Software Communication Architecture's Core Framework Base Device and Framework Control Interfaces [3, 50, 52, 63]

Figure 2.9 illustrates all of the Framework Services Interfaces and some of the relevant Framework Control Interfaces, and indicates the relationship between these interfaces. The *File*, *FileSystem*, and *FileManager* interfaces provide the framework with the ability to access and manage files both locally and in a distributed environment. The *DeviceManager* interface creates a *FileSystem* for each local hardware domain, all of which are managed by the *FileManager* launched by the *DomainManager*. (See [50, 52, 63])

The Domain Profile, which is also part of the SCA's Core Framework, is a collection of files which contain all the necessary information and instructions needed to manage and control the framework. The files are written in Extensible Markup Language (XML) according to prescribed formats contained within the SCA specification. The files include such information as a list of available hardware components and their

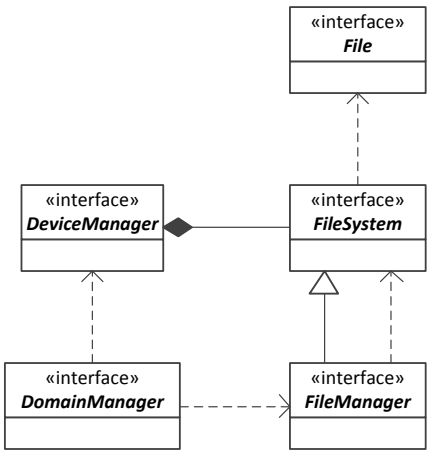


Figure 2.9: The Software Communication Architecture's Core Framework Services and Framework Control Interfaces [3, 50, 52, 63]

capabilities, a list of available waveforms along with the components needed to create them, a description of each software component and its required resources, and the configuration for the framework when initially launched. (See [50, 52, 63])

2.2.2.4 The OMG Platform Independent Model and Platform Specific Model for Software Radio Components

The OMG's Platform Independent Model (PIM) and Platform Specific Model (PSM) for Software Radio Components Specification [65] redefines the SCA specification using an extension of the Unified Modelling Language (UML) called UML Profile for Software Radio. The UML Profile for Software Radio adapts UML for use within the SDR environment by including software defined radio specific definitions [65,66]. Although the specification does include a few additional functions beyond those of the SCA, they are only provided to resolve ambiguities in the SCA specification by enhancing the abstraction between waveform applications and the underlying hardware [3, 65, 66].

The primary purpose of the UML Profile for Software Radio is that it decouples the SDR functionality from specific implementation technologies such as CORBA. Developers are free to design their software components according to a PIM, and then implement the PIM using specific technologies, resulting in a PSM such as the SCA. (See [3, 65, 66])

2.2.3 The ETSI, E^{2}RII, and E3 Functional Architecture for the Management and Control of Reconfigurable Radio Systems

The End-to-End Efficiency (E3) project was a European reconfigurability research initiative that formed part of the European Union's (EU) 7th Framework Programme (FP) during 2008 and 2009. E3 built upon work performed by previous European reconfigurability projects, specifically the End-to-End Reconfigurability (E^{2}R) project, and its successor, the End-to-End Reconfigurability phase 2 (E^{2}RII) project. (See [67, 68])

The E3 Functional Architecture (FA) for the Management and Control of Reconfigurable Radio Systems is one of the primary contributions made by the E3 project [57]. The purpose of this functional architecture, which is referred to simply as the E3 FA, is to provide a framework for the management and optimisation of all radio resources and spectrum utilisation within a RAN through the use of reconfigurability.

In 2009 the ETSI Technical Committee for RRSs, after examining the E3 FA, published Technical Report 102.682 [43] which presented the ETSI's own FA for managing RRSs. The ETSI FA is virtually identical to the E3 FA, with only one additional component in the E3 FA when compared with the ETSI FA. The following subsections present the requirements upon which the E3 FA is based, and an overview of the E3 FA, including its logical components, interfaces, and operations.

2.2.3.1 E3 Functional Architecture Requirements

The E3 FA is based on a set of operational scenarios which the FA is expected to support, which in turn provide a set of requirements for the architecture. These requirements are virtually identical to those published by ETSI as requirements for their FA, with the only exception being that the ETSI FA does not include the requirement for the network to autonomous manage itself, referred to as "self-x" by the E3 project. (See [57, 67, 68])

The E3 FA's operational scenarios can be divided into three categories relating to spectrum management, cognitive radio, and autonomous network management (self-x) of all network elements within a RAN [43, 57]:

- **Spectrum Management**: The FA must be capable of supporting dynamic spectrum allocation and management between different RATs deployed in separate cells and assigned to different users. Secondary use of spectrum, either by the FA itself or opportunistically by an unlicensed third-party, must also be supported.

- **Cognitive Radio**: The FA must support cognitive radio functionality which determines when to perform reconfiguration of network elements, based on user preferences, network capacity, QoS requirements, network element capabilities, and available RATs. In addition, such functionality must be provided transparently without the user being aware of the heterogeneous nature of the RAN, and a CPC must be used to support all Cognitive Radio functions.

- **Self-x**: The FA must provide for the autonomous management of network elements, thereby enabling such elements to react to and resolve a range of predefined situations. Such functionality is inherently geared towards the rapid short-term resolution of any potential network operational problems, or the short-term optimisation of network resource utilisation.

2.2.3.2 E3 Functional Architecture Overview

The E3 FA, presented in Figure 2.10, contains a set of logical components which are divided between each RMD, each RBS, and the heterogeneous RAN as a whole. Instances of the Joint Radio Resource Management (JRRM) and the Configuration Control Module (CCM) exist in both the RMD as well as the RAN, while the Dynamic Spectrum Management (DSM), Dynamic Self-Organising Network Planning and Management (DSONPM), and Self-x for RAN exist solely in the RAN. (See [43, 57, 67, 68])

The DSM is responsible for managing and allocating spectrum over the medium- and long-term to all RATs within the RAN. Its purpose is to ensure the efficient utilisation of the spectrum, by maximising spectrum reuse between RATs deployed on network elements within the RAN, while minimising interference between them. The DSM contains all the information relating to regulatory spectrum policies and current spectrum assignments, and utilises this information to perform spectrum allocations to different RATs. The DSM is also capable of administering spectrum sharing and leasing, and can trade spectrum with other DSM instances in different RANs. (See [43, 57, 67, 68])

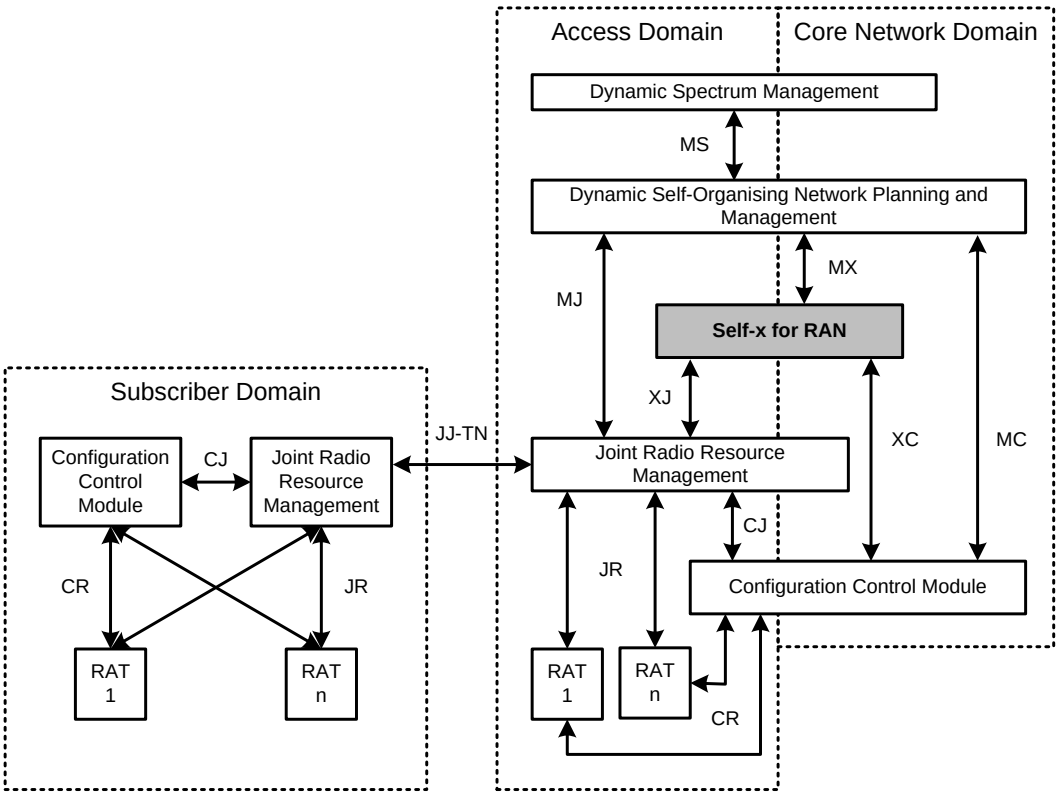


Figure 2.10: The E3 Functional Architecture for the Management and Control of RRSs [43, 57, 67, 68]

The DSONPM makes medium to long-term decisions regarding the reconfiguration of network elements within the RAN. The DSONPM receives context information from the JRRM containing the current status of each RAT and network element, the capabilities of each network element, preferences of users, QoS requirements, and the distribution of traffic between different RATs within each RBS. This information, in combination with the spectrum allocation decisions provided by the DSM, enables the DSONPM to determine whether any reconfiguration of network elements should take place in order to optimise the performance of the entire RAN. This decision making capability is enhanced through the inclusion of algorithms that learn from the results of previous decisions. (See [43, 57, 67, 68])

The JRRM is responsible for the joint management of all radio resources throughout the RAN. It collects context information from RATs within each network element, and provides this information to the DSONPM. The JRRM focuses on the short to medium-term management of radio resources, and adjusts the parameters of RATs and network elements within the RAN so as to achieve optimal utilisation of their

radio resources. The JRRM is not responsible for reconfiguring network elements, and can only adjust the configuration of such elements. In the event that the JRRM determines that a reconfiguration may be required, the JRRM alerts the DSONPM so that it can determine the proper course of action. The RAN's JRRM is also responsible for gathering context information from each RMD's JRRM, and relaying reconfiguration decisions and spectrum allocations to these devices via a CPC. (See [43, 57, 67, 68])

The autonomous management of operational tasks is performed by the Self-x for RAN logical component. This component is not present in the ETSI FA, and its functionality could theoretically be performed by the DSONPM; however whilst the DSONPM focuses on medium to long-term reconfiguration decisions, the Self-x for RAN is designed to manage the operations of network elements in the short-term. Examples of where autonomous management of operations is used include handover parameter optimisation, failure of network elements, and load balancing between elements. Note that the Self-x for RAN requires that network operators establish predefined procedures for handling these situations, and the primary goal is to ensure rapid predictable resolution of network problems. The Self-x for RAN acquires context information from the JRRM, and performance requirements and policies from the DSONPM. Instructions to adjust network element parameters or reconfigure such elements are sent by the Self-x for RAN to the JRRM and CCM respectively. (See [57, 67])

The operations that reconfigure network elements, and those that adjust their parameters, are performed by the CCM based on instructions from the DSONPM, JRRM, and Self-x for RAN. The reconfiguration of a RAT within an RBS can only be performed after the JRRM has transferred all existing connections to another RBS. (See [43, 57, 67, 68])

The E3 FA specification also contains a detailed description of each of the reconfiguration management and control operations that may be performed by the FA, and provides an information model which describes the manner in which information within the logical components is defined and managed. A detailed analysis of FA's operations and information model is outside the scope of this thesis, and can be found in [57].

2.2.4 The IEEE 1900.4 Standard

In 2005 the IEEE initiated development of a new series of standards designed to focus on issues relating to next generation radio systems and advanced spectrum management techniques. These standardisation efforts are currently managed by Standards Coordinating Committee 41 (SCC41). (See [69])

To date the SCC41 has published several standards, including the 1900.1 standard [4] mentioned in Section 2.1, which contains definitions and terminology relating to emerging wireless systems and spectrum management.

The 1900.4 standard published in 2009 [56], contains a functional architecture for the management and control of RRSs. This architecture, like the functional architecture developed by the E2RII and E3 European Framework Projects, focuses on using reconfigurable radio systems within heterogeneous telecommunications networks to optimise network capacity and achieve QoS, and to ensure efficient utilisation of radio resources and available spectrum. (See [69, 70])

The following subsections examine the use cases and requirements upon which the standard is based, and provide an overview of the architecture with its logical components, interfaces, information model, and management procedures.

2.2.4.1 IEEE 1900.4 Use Cases and Requirements

The development of the 1900.4 standard is based on a set of use cases that describe the scenarios in which the architecture needs to operate. These use cases are: Dynamic spectrum assignment, dynamic spectrum sharing, and distributed radio resource usage optimisation. (See [56, 69, 70])

The dynamic spectrum assignment use case refers to a scenario in which spectrum allocations are dynamically assigned to different RATs within the same or different RANs. This opens up the possibility of both spectrum sharing and temporary spectrum swapping between RATs and RANs. (See [69, 70])

In the case of dynamic spectrum sharing, the relevant portions of the spectrum are statically assigned to one or more RATs within their respective RANs; however RATs can choose to share the available spectrum as the need arises. An example of such a situation is when a RAT in one RAN opportunistically makes use of spectrum

already assigned to another RAT in a different RAN, during those times when the latter does not require the spectrum. (See [69, 70])

Lastly, the distributed radio resource usage optimisation use case is designed to assist legacy RATs to optimise their radio resource usage and improve their QoS. Spectrum assignments are fixed, and reconfiguration of access network equipment, such as Base Stations, is not supported. Mobile Terminal reconfiguration is possible, but the decision to perform any reconfiguration is based on information and policies supplied by the access and core networks. (See [69, 70])

Using the three use cases described above, the SCC41 derived a set of requirements for the 1900.4 architecture. These requirements are encapsulated within the three stages involved in any reconfiguration process [56, 70]:

- **Context Awareness**: Both the mobile terminal and base station collect, analyse and store context information. The base station's context information contains all radio resource optimisation objectives, the base station's capabilities, and measurements of the wireless environment and the base station's internal state.
- **Decision Making**: The RAN must manage the radio resources of the network as a whole, and provide each terminal with a framework within which it must operate. Decision making is performed by Resource Managers in the RAN and each terminal using a policy-based management framework.
- **Reconfiguration**: Both the mobile terminal and base station must possess the capability to reconfigure themselves based on the decisions made by their respective Resource Managers.

2.2.4.2 IEEE 1900.4 Architecture Overview

An illustration of the 1900.4 Logical Architecture is presented in Figure 2.11. The architecture is composed of seven logical components, divided between three functional domains representing the areas containing subscribers' terminals, access network equipment, and the core network. The architecture also specifies the six interfaces that connect the logical components.

As was briefly mentioned previously, the architecture utilises a policy-based management framework which contains two types of policies: a spectrum assignment

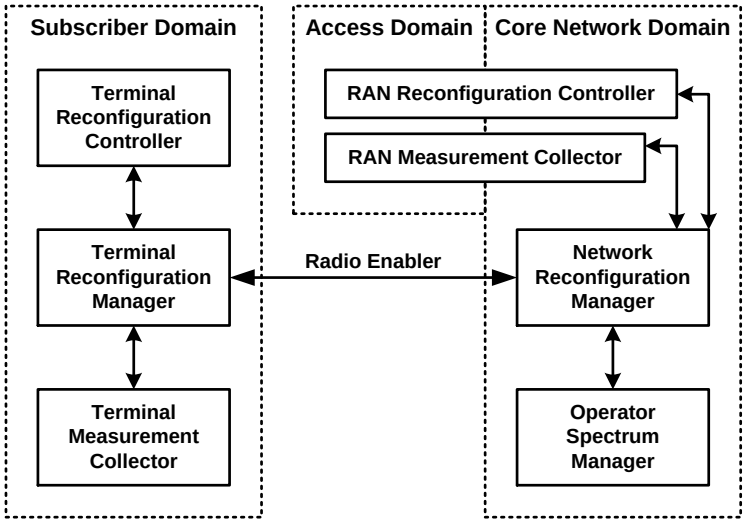


Figure 2.11: The IEEE 1900.4 Logical Architecture [56, 69, 70]

policy and a radio resource selection policy. Spectrum assignment policies provide NOs with the ability to efficiently manage their spectrum allocation and ensure that regulatory conditions are met. Radio resource selection policies are generated by the core network and provide the constraints within which the terminal may reconfigure itself. (See [56, 69, 70])

At the core of the 1900.4 architecture lies the Network Reconfiguration Manager (NRM). The NRM is responsible for managing the radio resources for the RAN and all terminals. The RAN Measurement Controller provides the NRM with context information regarding the RAN, whilst the Operator Spectrum Manager provides the NRM with spectrum assignment policies. Using this information, the NRM must generate a set of radio resource selection policies for all connected terminals, and constantly evaluate these policies in an effort to optimise them. (See [56, 69, 70])

The RAN Reconfiguration Controller is responsible for performing a reconfiguration of the RAN, based on decisions made by the NRM. The NRM must also continuously ensure that all reconfiguration decisions obey the guidelines set out in the current spectrum assignment policies. (See [56, 69, 70])

The Terminal Reconfiguration Manager (TRM) performs similar functions to the NRM within the terminal. The TRM also collects context and spectrum information, supplied by the Terminal Measurement Collector, and determines whether a reconfiguration should take place. Any reconfiguration must be performed within the guidelines stipulated by the radio resource selection policy provided by the NRM. The Terminal Reconfiguration Controller implements any reconfiguration decision taken by the TRM. (See [56, 69, 70])

The TRM and NRM are connected by means of a Cognitive Pilot Channel called the Radio Enabler. The Radio Enabler provides the TRM and NRM with a means of exchanging radio resource selection policies and context information. The exact method used to implement the Radio Enabler is not specified, but existing or dedicated RATs could be used depending on available radio resources. (See [56, 69, 70])

Figure 2.12 illustrates an enhanced version of the IEEE 1900.4 Architecture, indicating the functions performed by the TRM and NRM.

The 1900.4 standard contains an information model designed to assist developers in implementing the architecture in software. The information model is based on an object-oriented approach, and is presented using high-level class diagrams that represent the architecture's logical components, and their functions and configurations. A complete description of the model is outside the scope of this thesis, and may be viewed in the IEEE 1900.4 standard [56].

2.2.5 Commercial Base Station Architectures

There are currently two major commercial standards for radio base station architectures, namely the Open Base Station Architecture Initiative (OBSAI) [71], and the Common Public Radio Interface (CPRI) [72]. The purpose of each of these standards is to divide individual base stations into two logical components, one of which performs each RAT's baseband signal processing, and the other which contains the analogue radio frequency (RF) components required for transmitting and receiving signals in the wireless environment [40].

The CPRI standard limits itself to specifying the interface between the baseband and RF logical components, referred to by the standard as the Radio Equipment Control (REC) and Radio Equipment (RE) respectively, whereas the OBSAI standard specifies several additional logical components which assist in the management and control of its baseband and RF logical components.

The CPRI standard maps the Physical and Data Link Layers' functionality onto the REC and RE for a specified set of RATs; these represent the 3GPP's Universal Terrestrial Radio Access Network (UTRAN) and evolved UTRAN (E-UTRAN) standards, and the IEEE's 802.16 (WiMAX) standard. Although the CPRI does support the ability to implement these RATs in software, it does not contain any

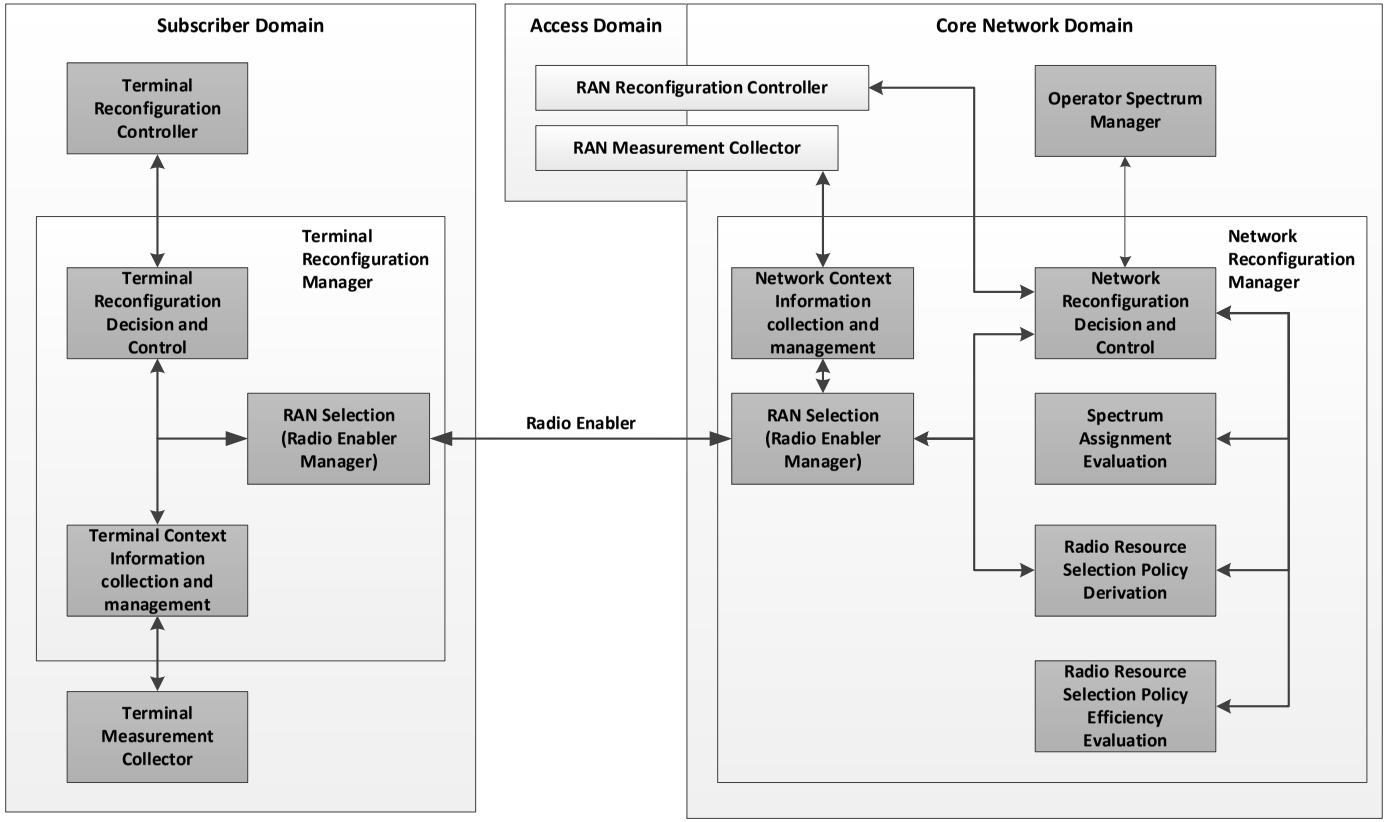


Figure 2.12: The IEEE 1900.4 Functional Architecture [56,69,70]

of the functionality required to manage and control the dynamic reconfiguration of the RATs' functionality in an individual base station or in the RAN as a whole. It also does not specify any standardised interfaces for radio application development, implementation, and deployment on reconfigurable hardware platforms. As such the CPRI cannot be considered to be a RRS, however due to its broad acceptance amongst network equipment vendors, it is important to mention and discuss this standard within the context of RRSs. (See [72])

Figure 2.13 describes the OBSAI's logical architecture, which consists of four main logical components, namely the Transport block, the Baseband block, the RF Block, and the Control and Clock block, all of which are connected via a set of standardised interfaces.

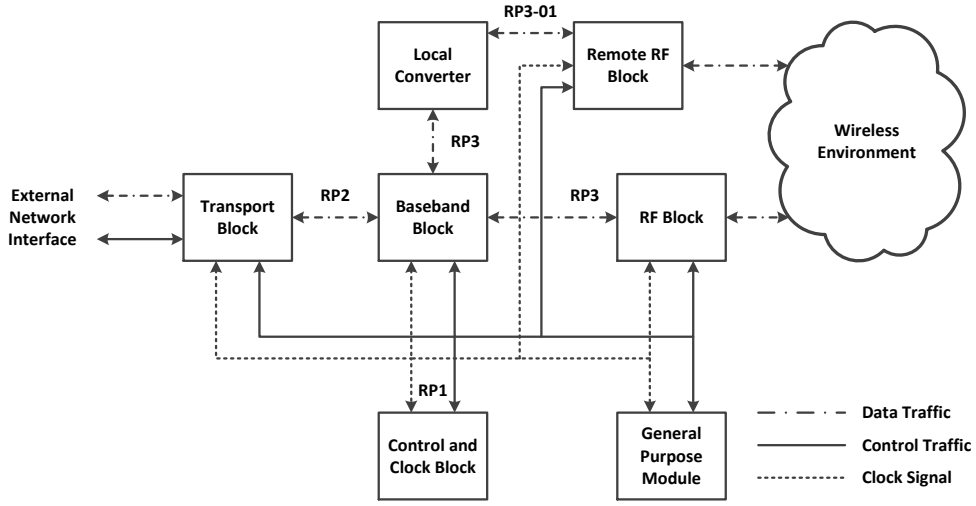


Figure 2.13: The OBSAI Architecture [40, 71]

The OBSAI's Transport block is responsible for connecting the base station to an external network. This Transport block is capable of supporting a variety of networking protocols, including Asynchronous Transfer Mode (ATM), Internet Protocol (IP), Ethernet, Synchronous Digital Hierarchy (SDH), and Synchronous Optical Network (SONET). The RP2 interface connects the Transport block to the Baseband block, and carries data destined for transmission by the base station as well as data received by the base station which must be conveyed to the core network. (See [71])

The Baseband block implements each RAT's baseband signal processing, and is capable of supporting multiple RATs simultaneously. Although the OBSAI standard

does provide a list of supported RATs, including GSM, W-CDMA, CDMA2000, HSPA, LTE and WiMAX, the architecture is capable of supporting any RAT as long as such functionality is implemented by the Baseband block. The Baseband block communicates with the RF Block via the RP3 interface, which transports digital baseband signals which are transmitted and received by the base station. (See [71])

The RF block contains the electronic components required to convert the digital baseband signal into an analogue signal for transmission, and vice versa. It also contains the analogue frequency components needed to perform signal filtering, adjust transmission power, synchronise received signals, and control and interface with antennae. The OBSAI standard also includes a logical component known as the Local Converter and the RP3-01 interface, which together provide very similar functionality to the CPRI and enable the physical separation of the Baseband and RF blocks. (See [71])

Management and control of the base station and its logical components is performed by the Control and Clock block. This block monitors the status of all components and their performance, and using this information it adjusts the parameters of the architecture's other interfaces, as well as the configuration of the RATs currently implemented, to ensure optimal use of radio resources and also to ensure that connections provide acceptable levels of QoS. The Control and Clock block is also responsible for providing the clock signals used by other blocks to synchronise their interfaces and internal operations. (See [71])

Although OBSAI does not contain any procedures or functions to perform reconfigurability, the interfaces provided by the standard are capable of supporting dynamic reconfiguration, provided that the required functionality is implemented within the architecture's blocks [40]. The standard also contains a logical component called the General Purpose Module, which can perform any function which is required by the base station, which is not supported by any of the other blocks [71]. This General Purpose Module could be used to perform the necessary functions required to perform dynamic reconfigurability within an OBSAI-compliant base station.

2.2.6 Reconfigurable Protocol Stacks

Until fairly recently, RATs were commonly modelled using the ITU's Open Systems Interconnection (OSI) model [73], which was first published in 1983. Since the OSI

model's introduction, many new communication technologies and techniques have been developed, and many modern communication systems are no longer able to fit within the model's rigid layering system [74–76].

Stratification, or layering, is a specialised form of component-based software modelling used to partition and encapsulate RAT functionality so that each component can be developed and implemented independently, whilst still ensuring that their combination results in a complete communications system. Stratification is made possible by the fact that many RATs share the same functionality and underlying technologies.

Reconfigurable Protocol Stacks (RPS) generalise the principle behind stratification by decomposing RATs into their fundamental components, and encapsulating the resulting functions within software components [47, 77–81]. A RRS can then use these components to reconstruct the original RATs during reconfiguration operations.

The primary benefits associated with using RPSs in RRSs are as follows [47, 77–81]:

- **Reduced development time and costs**: Radio application developers can significantly reduce the amount of time and effort required to create new RATs, as many of the software components used to implement older RATs can be reused.
- **Improved reliability**: As software components are reused when developing new RATs, faults or inefficiencies are more likely to be either avoided or quickly detected and resolved, thereby improving the reliability of individual components and RAT implementations.
- **Increased Flexibility and Scalability**: As RPSs do not impose any limitations on the structure or number of software components used to construct RATs, any legacy, current, or future RAT can be developed and implemented using RPSs.
- **Incremental standardisation and upgrading of RATs**: Instead of developing and then standardising an entire RAT, standards bodies could publish specifications for individual RPS components and schemas for assembling those components into a specific RAT. This approach would increase the rate at which RATs are standardised, and would enable incremental changes to RAT standards which could be rapidly implemented by downloading the software OTA.

- **Runtime Reconfiguration of RATs**: RPSs facilitate and improve a RRS's ability to reconfigure existing RATs or construct new RATs at runtime.

The principle behind RPSs is not new and has been used in many communication system simulation tools to implement the Physical layers and DLLs of RATs. Some examples of these packages include the open source GNU Radio project [82], the Network Simulator (NS) series [83,84], and the Simulink dynamic system modelling tool from Mathworks [85]. The RPS approach to creating radio applications has also been utilised in several RRSs, including the SCA and the OMG's UML for software radio. It should be noted that none of these RRSs or communication system simulation tools are capable of dynamically reconfiguring their protocol stacks at runtime.

The following subsections present the two different modelling approaches for RPSs, the logical components and operations used to manage and control RPSs, the implementation issues involved is developing RPS systems, and an example of a RPS developed by the European WINNER research project.

2.2.6.1 Reconfigurable Protocol Stack Modelling Methodologies

There are two different approaches which may be used to model and encapsulate RAT functionality within software components in a RPS. Both of these approaches require the identification of functions which are common to many or all RATs, and of those functions which are unique to individual RATs.

The Adaptable Protocol Stack approach implements specific RATs by combining generic layers which contain common RAT functions with customised extension layers containing functions unique to individual RATs [77,81]. Figure 2.14 describes the Adaptable Protocol Stack approach and illustrates how the construction of new RATs, and the adjustment of existing RATs, is accomplished by varying which extension layers are combined with each generic layer. The Adaptable Protocol Stack approach is based on the premise that common RAT functions can always be implemented in the same layers in different RATs; however this is often not the case as many RATs implement common functions in a different order to one another.

The second approach to modelling and implementing RPSs is known as the Composable Protocol Stack approach. This approach decomposes a RAT into its atomic functions, both common and RAT specific, and encapsulates each one in its own

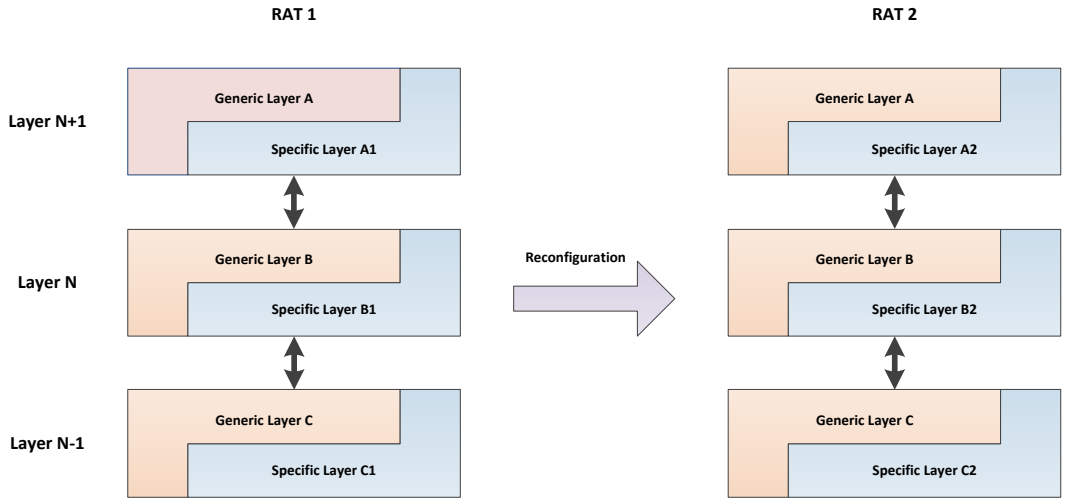


Figure 2.14: The Adaptable Protocol Stack approach to modelling and implementing Reconfigurable Protocol Stacks [77, 81]

software component [77, 81]. The combination of these software components in particular configurations results in the implementation of a specific RAT. The Composable Protocol Stack approach, illustrated in Figure 2.15, is significantly more flexible and scalable than the Adaptable Protocol Stack approach, as functions are not restricted to specific layers and the structure of the stack itself is not limited to any predefined configuration.

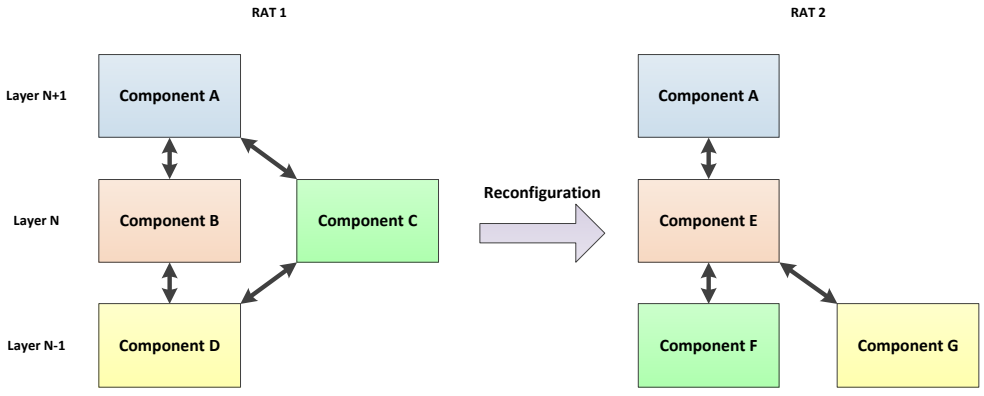


Figure 2.15: The Composable Protocol Stacks approach to modelling and implementing Reconfigurable Protocol Stacks [77, 81]

The Adaptable Protocol Stack approach can be implemented using software inheritance, whereby each extension layer is a subclass of its associated generic layer class. The Composable Protocol Stack approach can be implemented either through the use of inheritance, or by using parameterised software modules whose functionality can be adapted by changing each module's parameters. Example of possible parameters include simple software variables, software switches which turn functionality on and off, and customised interfaces to connect the module to an external input. (See [76, 80])

2.2.6.2 Managing Reconfigurable Protocol Stacks

A RPS requires a management system to perform the operations involved in generating, adapting, and destroying RAT implementations. The E3 Functional Architecture and the IEEE 1900.4 standard both provide some of the functionality which could be used to manage a RPS, however due to the flexibility and complexity associated with constructing a RAT using the RPS approach, there are a number of management and control operations which are not covered by either system. The following lists all of the functionality required to manage and control a RPS [79–81, 86]:

- Each implemented RAT, and all of its software components, must be monitored in order to obtain status and performance information.
- Based on predefined objectives, or in an effort to optimise radio resource utilisation, a decision must be taken regarding when the reconfiguration of a RAT should take place. This decision could involve the construction and implementation of a new RAT, the adjustment of an existing RAT's parameters or software components, or the complete destruction and removal of an existing RAT implementation.
- Prior to a reconfiguration decision being implemented, each software component must be tested to ensure that it operates correctly, and the configuration specifications which define how the RAT is to be implemented must be validated to ensure they are feasible.
- Dynamic runtime reconfiguration operations require that the transition between RATs is achieved seamlessly, and that existing connections are transferred to new RAT implementations in cases where existing RAT implementations are destroyed and removed.

- During the normal operation of a RPS, the connections and flow of traffic between software components must be regulated to ensure that data is processed timeously and is not lost.

In addition to the management functionality presented above, a RPS also contains a secure operating environment which provides the facility to store software components and protocol stack configurations, and interfaces between the implemented RATs and the rest of the communication system [3, 6, 47]. Figure 2.16 illustrates a RPS's operating environment and management system.

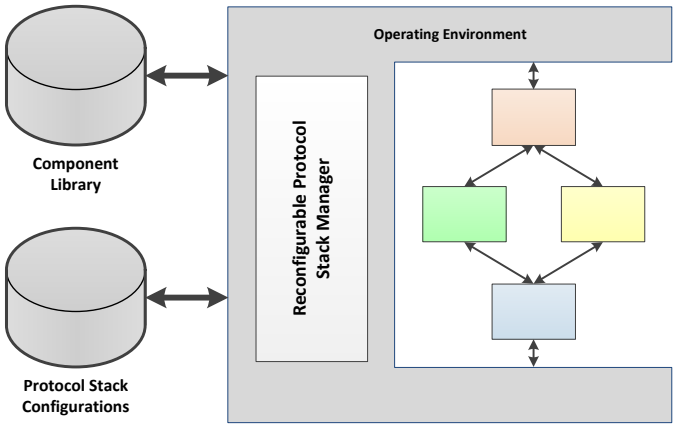


Figure 2.16: Reconfigurable Protocol Stack management and operating environment [3, 6, 47, 77, 79–81, 86]

2.2.6.3 Reconfigurable Protocol Stack Implementation Issues

The implementation of a RPS presents a number of issues, or design decisions, which must be resolved before a RPS can be realized. The first issue involves identifying the method which is used to decompose RATs and identify their fundamental components. Without a standardised set of rules, different developers could decompose each RAT differently, resulting in software components containing overlapping functionality and decreased radio application interoperability and portability [79,87].

The manner in which software components are modelled is another issue. This concerns the definition of the interfaces which exist between components, the manner in which a component's functionality is adjusted, and the method by which status and performance information is relayed to the management system [76, 80, 81].

The structure of the RPS determines whether it is capable of supporting multiple simultaneous RAT instantiations and whether it limits the manner in which software components are connected. In addition, an RPS must specify the format of protocol stack configuration plans which unambiguously define how a RAT is constructed and implemented [77, 79, 80].

Finally, the performance of the RPS is directly affected by the manner in which each software component is executed at runtime. There are two possible options, namely: threading per layer or component, and threading per message [77, 79, 88]. Threading per layer or component implements each software component in its own thread, which improves the rate at which individual functions are performed but also results in an overall decrease in performance as each message must be transferred between threads for processing [88]. Threading per message treats each component as a function which performs some action on the message. This method is faster than the former, but its performance is directly linked to the number of messages simultaneously processed by the system and the total processing resources available [88].

2.2.6.4 The WINNER multi-mode protocol reference architecture

The Wireless World Initiative New Radio (WINNER) project was a RRS related research initiative which, like the E^2R, E^2RII, and E3 projects, formed part of the European Union's Framework Programmes. Its objective was to develop a new RAT which was capable of completely adjusting its functionality to suit any wireless environment or deployment scenario [89].

In 2004 the WINNER project published a specification for multi-mode protocol reference architecture [90]. This architecture, presented in Figure 2.17, is one of the few RRSs developed by researchers and standard bodies to contain both a RPS and a reconfiguration management system.

The WINNER architecture employs a hybrid modelling approach, which in turn incorporates aspects of both the Adaptable Protocol Stack and the Composable Protocol Stack approaches. The architecture defines four layers, namely the Radio Resource Control (RRC) layer, the Radio Link Control (RLC) layer, the Medium Access Control (MAC) layer, and the Physical (PHY) layer. Each layer is split into a generic sublayer and multiple RAT specific sublayers, which are depicted in Figure 2.17 with the "g" and "r$_x$" suffixes respectively. Software components which

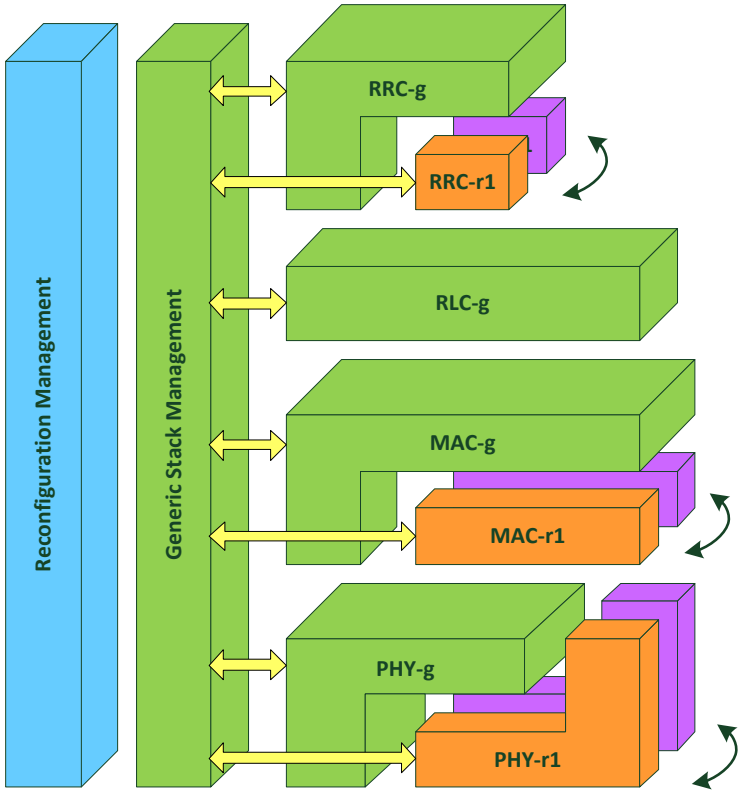


Figure 2.17: The WINNER multi-mode protocol reference architecture [5, 89–91]

implement RAT functionality are assigned to specific sublayers in the WINNER architecture based on the nature of the functionality they provide, and whether such functionality can be reused to implement numerous RATs or is unique to an individual RAT. (See [89–91])

The architecture's management system is subdivided into managers which administer individual layers, known collectively as the Generic Stack Management System, and a general Reconfiguration Management System which coordinates the reconfiguration operations of the entire RAN. The general Reconfiguration Management System is responsible for the efficient allocation of all radio resources, determining whether a protocol stack should be reconfigured, and allocating traffic flows to RATs based on their requirements and available radio resources. (See [5, 89–91])

A novel concept introduced by the model is the idea that some layers with the same functionality, but located in separate RATs, can be horizontally converged [5,89–91].

The implementation of such a concept would make it possible for many different RATs to converge at least some of their functionality, possibly even resulting in a single monolithic RAT. Unfortunately, the WINNER architecture does not specify in any significant detail how separate traffic flows can be identified so that they can be assigned to their respective RATs.

The WINNER architecture is unique due its combination of RRS management systems and RPSs, however the architecture is not detailed enough to ensure that separately developed implementations would be interoperable and it is also unduly restrictive due to its predefined layering scheme.

2.3 Requirements for a Converged Reconfigurable Radio System Architecture

The previous sections presented each of the major RRS research and standardisation initiatives, with a focus on their architectures and RRS related functionality. As each RRS initiative focuses on different outcomes when developing its systems, each RRS provides different levels of functionality and models its functions using a different methodology.

By analysing and comparing the functionality provided by each of the reviewed RRSs, it is possible to develop a complete list of functional requirements describing the functions which the CRRSA must support. These requirements are divided into three categories, namely the RRS's Operating Environment, Reconfiguration Management system, and Radio Applications.

2.3.1 Operating Environment Functional Requirements

An RRS's operating environment provides the supporting infrastructure which enables the RRS's management system and radio applications to successfully complete their tasks. Table 2.1 presents the three functional requirements for an RRS's operating environment and briefly outlines if, and how, each of these functions are implemented in each of the reviewed RRSs. The following subsections provide a more detailed explanation of each of these functional requirements:

Table 2.1: Operating Environment Functional Requirements for Reconfigurable Radio Systems

	ETSI SDR	ETSI RBS	SCA	OMG	E3 FA	1900.4	OBSAI	WINNER
Interface to the Protocol Stack's Upper Layers and backhaul to the Core Network	**Yes**, via the Multi-Radio Access Interface. Does not provide any details regarding implementation	**Yes**, via a logical component called the Backhaul Management. Does not provide any details regarding implementation	**Undefined**, although the requirement is mentioned, the system contains no formally defined interface to the Protocol Stack's Upper Layers and backhaul to the Core Network	**Undefined**, although the requirement is mentioned, the system contains no formally defined interface to the Protocol Stack's Upper Layers and backhaul to the Core Network	**Undefined**, although the Dynamic Self-Organising Network Planning and Manager logical component is capable of selecting which backhaul option to utilise, there is no formally defined interface to the Protocol Stack's Upper Layers and backhaul to the Core Network	**Undefined**, although the requirement is mentioned, the system contains no formally defined interface to the Protocol Stack's Upper Layers and backhaul to the Core Network	**Yes**, a well-defined logical component called the Transport block provides functionality to interface with core network	**Undefined**, although the requirement is mentioned, the system contains no formally defined interface to the Protocol Stack's Upper Layers and backhaul to the Core Network
Interface to SDR hardware	**Yes**, via two interfaces called the Unified Radio Application Interface and Reconfigurable Radio Frequency Interface. Does not provide any details regarding implementation	**Yes**, via a standardised interface, but does not provide any details regarding implementation	**Yes**, but only for CORBA enabled signal processing components. Other interfaces designed to interwork with the SCA have been separately developed	**Yes**, via standardised well-defined technology independent interfaces. Such interfaces will need to be mapped to specific technologies during implementation	**Undefined**, system does not contain any formally defined interface to SDR hardware	**Yes**, via standardised well-defined logical components with interfaces.	**Yes**, the RF block provides the functionality to transmit and receive analogue signals, and translate them from/to digital baseband signals. Specific implementation details for any block in the OBSAI architecture are not provided	**Undefined**, although the requirement is mentioned, the system contains no formally defined interface to SDR hardware
Secure Communications and Operating Environment	**Undefined**, although the requirement is mentioned, the system contains no formally defined security mechanisms	**No**, system does not contain any security related functions	**No**, system does not contain any security related functions	**Yes**, the contains logical components called Network Applications that provide security functionality	**No**, system does not contain any security related functions	**No**, system does not contain any security related functions	**Yes**, a logical component called the Transport block provides security functionality	**Yes**, provided by general protocol stack management functions in the reconfiguration management plane. Does not provide any details regarding implementation

2.3.1.1 Interface to the Protocol Stack's Upper Layers and backhaul to the Core Network

An RRS must provide the functionality required to connect the RRS's management system and radio applications to the rest of the communications network. RRS management traffic needs to flow between the core network and RAN, and between the RAN and subscribers' mobile devices, in order to perform RRS management operations such as OTA software downloads, DSA, and obtaining context information for use in making reconfiguration decisions. In addition, radio applications also need to be able to connect to the traditional mobile software applications running on subscribers' mobile devices, and to the RAN's backhaul system so as to convey mobile service-related data traffic to and from the core network.

This functionality can be implemented via the creation of a standardised interface which enables the RRS management system and Radio Applications to connect to the Network layers in subscriber mobile devices and the RAN. The requirement for an interface of this nature is recognised by all the reviewed RRSs; however only the commercial OBSAI base station standard provides the specification for such an interface via its Transport block.

2.3.1.2 Interface to SDR hardware

The second functional requirement for an RRS's operating environment is to provide a standardised interface for the RRS management system and radio applications to access and utilise the signal processing components on SDR hardware platforms. The level of abstraction required between a RRS's hardware and software is a point of contention amongst RRS researchers. Depending on the focus of the RRS, radio applications and other software components may be tightly or loosely coupled with each of the SDR hardware platform and its components.

The OMG's PIM for Software Radio Components and the IEEE's 1900.4 standard both provide technology independent methods for modelling hardware components or devices. In contrast, the SCA defines specific CORBA interfaces which can be used to interface software components and radio applications with CORBA-enabled signal processing hardware. Some RRSs, such as the E3 FA and the WINNER RPS system, consider such an interface to be outside their scope and do not provide a formal specification for the interface.

Ideally an SDR hardware interface should be technology independent and expose a standardised set of functions for use by the RRS's software components. This approach ensures that the RRS is not limited by the capabilities of any specific implementation technology; however this is only possible if the functionality provided by all SDR platforms becomes standardised thereby guaranteeing the portability of RRS software.

2.3.1.3 Secure Communications and Operating Environment

The final RRS operating environment functional requirement is the need to provide tight, flexible, and future-proof security for RRS communications and the general operating environment. Despite the fact that the need for strong security mechanisms in RRSs is widely acknowledged, none of the reviewed RRSs provide a security solution which includes the functionality required to secure the transmission of new software components, new policies, and reconfiguration management messages, and also to secure the operating environment in order to prevent the direct installation of malicious software or unauthorised access to already installed software.

Traditional communication systems possess functionality which is static, and security mechanisms for such systems can be developed, implemented and then left largely untouched. A RRS's functionality is constantly changing however, which introduces new vulnerabilities and therefore new security threats. As a result there is a need for a flexible security system for RRSs which is able to adapt its own functionality to protect the RRS against new security threats as they arise.

2.3.2 Reconfiguration Management Functional Requirements

An RRS's management system provides all the necessary functionality to manage and control all of the operations and resources used to reconfigure the radio system, and it also specifies how reconfiguration operations are conducted. Table 2.2 presents the nine functional requirements for an RRS's management system and briefly outlines if, and how, each of these functions are implemented in each of the reviewed RRSs. The following subsections provide a more detailed explanation of each of these functional requirements:

Table 2.2: Reconfiguration Management Functional Requirements for Reconfigurable Radio Systems

	ETSI SDR	ETSI RBS	SCA	OMG	E3 FA	1900.4	OBSAI	WINNER
Cognitive Reconfiguration Capabilities	**Yes**, via a logical component called the Configuration Manager. Does not provide any details regarding implementation	**Yes**, contains an interface to trigger reconfigurations and software to manage the process. Does not provide any details regarding implementation	**No**, system does not contain any Cognitive Reconfiguration Capabilities	**No**, system does not contain any Cognitive Reconfiguration Capabilities	**Yes**, via a logical well-defined component called the Dynamic Self-Organising Network Planning and Manager	**Yes**, via a logical well-defined component called the Network Reconfiguration Manager	**Undefined**, although the system is capable of supporting Cognitive Reconfiguration Capabilities, this functionality is not formally defined	**Yes**, provided by general protocol stack management functions in the reconfiguration management plane, and logicial components that manage each layer. The reconfiguration procedure is broadly defined
Monitor and collect context information	**Undefined**, although indirectly implied by the inclusion of Cognitive Capabilities, the system does not formally define a context information collection function	**Yes**, via a logical component called Operations and Maintenance Does not provide any details regarding implementation	**No**, system does not provide a context information collection function	**Yes**, contains logical components dedicated to monitoring software components' status and performance	**Yes**, via a logical well-defined component called the Joint Radio Resource Manager	**Yes**, via a logical well-defined component called the RAN Measurement Collector	**Yes**, via a logical well-defined component called the Control and Clock block	**Yes**, provided by general protocol stack management functions in the reconfiguration management plane, and logicial components that manage each layer
Policy-based Management	**Yes**, via a logical component called the Mobility Policy Manager. Does not provide any details regarding implementation	**No**, system does not provide a Policy-based Management Solution	**No**, system does not provide a Policy-based Management Solution	**No**, system does not provide a Policy-based Management Solution	**Yes**, provides a complete policy-based management solution for spectrum allocation, radio resource management, RAT selection, and handover procedures	**Yes**, provides a complete policy-based management solution for spectrum allocation, radio resource management, and RAT selection	**No**, system does not provide a Policy-based Management Solution	**No**, system does not provide a Policy-based Management Solution
Validation of Software and Policies	**No**, system does not provide a software and policy validation function	**No**, system does not provide a software and policy validation function	**Yes**, via the well-defined TestableObject interface	**Undefined**, although no formal validation systems are defined, the system does lend itself to performing validation operations due to its well-defined interfaces	**No**, system does not provide a software and policy validation function	**No**, system does not provide a software and policy validation function	**No**, system does not provide a software and policy validation function	**No**, system does not provide a software and policy validation function
Dynamic Runtime Reconfiguration of RATs	**Yes**, via a logical component called the Configuration Manager. Does not provide any details regarding implementation.	**Yes**, contains the software and logical components to perform reconfiguration at runtime and perform vertical handover if required	**No**, system does not perform the Dynamic Runtime Reconfiguration of RATs	**Yes**, contains interfaces to that assist in the dynamic reconfiguration of radio applications, and hence RATs	**Yes**, via logical well-defined components called the Dynamic Self-Organising Network Planning and Management, the Joint Radio Resource Management, the Self-X for RAN, and the Configuration Control Module	**No**, system does not formally define the procedures or functions needed to perform the Dynamic Runtime Reconfiguration of RATs	**Undefined**, although the system is capable of supporting Dynamic Runtime Reconfiguration of RATs, this functionality is not formally defined	**Yes**, provided by general protocol stack management functions in the reconfiguration management plane, and logicial components that manage each layer. The reconfiguration procedure is broadly defined
Joint Radio Resource Control and Inter-RAT traffic scheduling	**Yes**, via two logical components called the Resource Manager and Flow Controller. Does not provide any details regarding implementation	**Yes**, via a logical component called the Radio Resource Management. Does not provide any details regarding implementation	**Partly**, although does contain the functionality required to divide available processing resources amongst radio applications and their software components, it does not possess the ability to jointly manage radio resources or schedule traffic between RATs	**No**, system does not possess the ability to jointly manage radio resources or schedule traffic between RATs	**Yes**, via a logical well-defined component called the Joint Radio Resource Manager	**Yes**, via a logical well-defined component called the Network Reconfiguration Manager	**Yes**, via a logical well-defined component called the Control and Clock block	**Yes**, provided by general protocol stack management functions in the reconfiguration management plane, and logicial components that manage each layer
Dynamic Spectrum Access	**Yes**, via a logical component called the Multi-Radio Controller. Does not provide any details regarding implementation	**Yes**, via a logical component called the Spectrum Management. Does not provide any details regarding implementation	**No**, although the frequencies assigned to individual radio applications can be adjusted, this cannot be performed dynamically	**No**, although the frequencies assigned to individual radio applications can be adjusted, this cannot be performed dynamically	**Yes**, via a logical well-defined component called the Dynamic Spectrum Manager	**Yes**, via a logical well-defined component called the Operator Spectrum Manager	**Undefined**, although the system is capable of supporting Dynamic Spectrum Access, this functionality is not formally defined	**Yes**, provided by general protocol stack management functions in the reconfiguration management plane
OTA Download of Software Upgrades and Policies	**No**, system does not provide an OTA software download function	**Yes**, via a logical component called the Software Management. Does not provide any details regarding implementation	**No**, system does not provide an OTA software download function	**Yes**, via a logical component called the IO Facilities. Does not provide any details regarding implementation	**Yes**, via a logically well-defined component called the Configuration Control Module	**No**, system does not provide an OTA software download function	**No**, system does not provide an OTA software download function	**Yes**, provided by general protocol stack management functions in the reconfiguration management plane
Cognitive Pilot Channel Solution	**No**, system does not provide a CPC Solution	**Yes**, via a logical component called the Radio Resource Management. Does not provide any details regarding implementation	**No**, system does not provide a CPC Solution	**No**, system does not provide a CPC Solution	**Yes**, via the well-defined JJ-TN interface	**Yes**, via the well-defined Radio Enabler interface	**No**, system does not provide a CPC Solution	**No**, system does not provide a CPC Solution

2.3.2.1 Cognitive Reconfiguration Capabilities

The first functional requirement for a RRS management system relates to whether the system is actually able to perform reconfiguration operations, and more specifically whether such operations can be performed autonomously based on predefined objectives and context information. In other words the first functional requirement of an RRS management system is that it must contain a Cognitive Engine.

RRSs which focus on defining SDR Architectures, such as the SCA and OMG, and current commercial base station systems, such as the OBSAI, do not contain any form of Cognitive Radio capabilities, and they assume that new radio applications are loaded directly onto the system. Cognitive Radio functionality forms a core component of all the other reviewed RRSs.

The scope of a Cognitive Radio's reconfiguration responsibilities, the methods used, and the degree to which reconfiguration is carried out, differ drastically between RRSs. For example, the E3 FA and the IEEE 1900.4 system both contain logical components which determine when and how to perform reconfiguration operations for all network elements within the RAN, whereas the WINNER system splits this functionality between a global reconfiguration manager and separate local reconfiguration managers for each layer in the protocol stack.

In addition, apart from the standard reconfiguration operations involving the creation and destruction of software components, an RRS management system can, at the cost of increased complexity, also include the functionality required to adapt existing software components at runtime. This approach could be useful for managing radio resources, but as many existing RATs already perform this function, its inclusion in the RRS management system simply results in increased complexity in order to deliver redundant functionality.

2.3.2.2 Monitor and Collect Context Information

In order for a RRS management system to be able to perform reconfiguration operations, it requires context information to determine the state of the wireless environment, the performance of existing RATs, and QoS requirements for existing traffic flows. Although virtually all of the reviewed RRSs, with the exception of the SCA, provide the ability to monitor and collect context information, only the OMG UML for Software Radio, the E3 FA, and the IEEE 1900.4 system actually provide well-defined logical components and operations for performing this function.

2.3.2.3 Policy-based Management

An RRS management system needs a set of predefined objectives which can guide it when making reconfiguration decisions. The optimal approach to this problem is to utilise a policy-based management system, which delivers the benefit of enabling autonomous reconfiguration operations. RRS management policies can be used to specify individual network elements' functional and processing capabilities, targets for QoS provision and radio resource utilisation, regulatory spectrum allocation requirements, and the available radio applications together with which RATs they implement. An additional benefit is that it allows a network operator to create policies which make it possible to autonomously determine which services are provided to individual subscribers. Of all the reviewed RRSs, only the E3 FA and IEEE 1900.4 contain well-defined policy-based RRS management systems.

2.3.2.4 Validation of Software and Policies

Prior to the execution of reconfiguration operations, the software components and policies involved in such operations should be validated to check for any errors, malicious or unauthorised changes, and to ensure that the relevant network elements are capable of supporting the changes in their functionality. The OMG's UML for Software Radio is the only reviewed RRS to provide any form of software component and policy validation; however this functionality is limited to simple software testing operations designed to ensure that individual components work correctly.

2.3.2.5 Dynamic Runtime Reconfiguration of RATs

In order to be able to fully exploit the benefits offered by RRSs, it must be possible to execute reconfiguration operations at runtime without needing to shut down or reboot the relevant network elements. Such functionality also requires the RRS management system to ensure that existing RAT implementations aren't negatively affected by such operations, and that network connections already established via existing RATs are transferred to new RATs if required.

RPS-focused RRSs, such as the WINNER system, provide an increased level of flexibility when performing reconfiguration operations at runtime, as their component-based architecture enables individual software components to be instantiated and

then connected to each other and the RRS operating environment without affecting the operations of existing software components.

2.3.2.6 Joint Radio Resource Control, Inter-RAT Traffic Scheduling, and Dynamic Spectrum Access

Many RRSs, such as the E3 FA and the IEEE 1900.4 and WINNER systems, provide the capability to jointly manage the radio resources of all network elements within a RAN, and dynamically allocate spectrum to different RATs. This functionality affords the network operator an unprecedented level of freedom to ensure that the distribution of radio resources is being constantly adjusted and optimised for efficiency, and that spectral resources are only assigned to individual RATs based on their QoS requirements and traffic load.

2.3.2.7 OTA Download of Software Upgrades and Policies

The rapid and efficient distribution of RRS software components and policies for use by an RRS management system is best achieved through the use of OTA download systems. This approach provides a network operator with the facility to simultaneously upgrade the functionality, reconfiguration instructions, and overall objectives of all the network elements in the RAN, and all subscribers' mobile devices, if required. As was discussed in the previous subsection, the downloading of new software and policies introduces new security threats not encountered in traditional communications systems. Encryption and certification mechanisms must therefore be included in any OTA download system. The OMG UML for software radio specification, the E3 FA, and the WINNER system all contain logical components designed to provide and manage this functionality.

2.3.2.8 Cognitive Pilot Channel Solution

As the functionality within reconfigurable network elements is constantly changing, the RRS management system must provide a facility for subscribers' mobile devices to be able to determine the appropriate RAT to implement, so as to be able to establish a connection with network elements in the RAN. This facility, introduced earlier in this chapter, is called a Cognitive Pilot Channel.

Despite the obvious need for such a function, only the E3 FA and IEEE 1900.4 system actually contain a formal specification for a CPC interface. What is not specified however is whether such an interface should be established using a dedicated RAT, or via existing RATs which are also used to convey mobile data traffic. The advantage of the former approach is that it ensures that RRS management system messages are not delayed or lost due to congestion generated by mobile services data traffic. The disadvantage of the dedicated RAT approach is that it may not cater for legacy network elements which do not support reconfiguration operations. Thus the optimal approach is to utilise existing RATs to establish a CPC, and to gradually move to a dedicated RAT system as legacy devices are phased out.

2.3.3 Radio Applications Functional Requirements

Radio Applications in RRSs must be able to be developed and ported to different platforms, and to be able to implement any legacy, current, or future RAT. Table 2.3 presents these functional requirements and briefly outlines if, and how, each of these functions are implemented in each of the reviewed RRSs. The following subsections provide a more detailed explanation of each of these functional requirements:

2.3.3.1 Radio Application Development Framework

A Radio Application Development Framework provides developers with a formal specification for software components, and a set of standardised interfaces which connect the components to each other and to the rest of the RRS. Such a framework specifies the functionality which each software component is expected to provide, and the means and procedures by which such components can gain access to the services provided by the RRS's operating environment. The degree to which this framework is defined directly affects the portability of software components between systems and the interoperability of software components within the same system.

Virtually all of the reviewed RRSs provided some form of well-defined Radio Application Development Framework, but as one of the primary goals of both the SCA and the OMG's UML for software radio is software portability and interoperability, both systems contained extremely detailed specifications for software components and their interfaces.

Table 2.3: Radio Applications Functional Requirements for Reconfigurable Radio Systems

	ETSI SDR	ETSI RBS	SCA	OMG	E3 FA	1900.4	OBSAI	WINNER
Radio Application Development Framework	**Yes**, via the Unified Radio Application Interface. Does not provide any details regarding implementation	**No**, system does not provide a Radio Application Development Framework	**Yes**, via a series of well-defined interfaces based on CORBA	**Yes**, via a series of well-defined technology independent interfaces	**Yes**, via a series of well-defined technology independent interfaces	**Yes**, via a series of well-defined technology independent interfaces	**Yes**, but the logical component called the Baseband module only specifies standardised interfaces to other logical components	**Yes**, service access point interfaces are specified for software components used to contruct RAT specific radio applications
Support any RAT using Reconfigurable Protocol Stacks	**No**, system does not provide a Reconfigurable Protocol Stack solution	**No**, system does not provide a Reconfigurable Protocol Stack solution	**Yes**, Radio Applications for specific RATs are constructed from individual software components with standardised CORBA interfaces	**Yes**, Radio Applications for specific RATs are constructed from individual software components with standardised technology independent interfaces	**Yes**, Radio Applications for specific RATs are constructed from individual software components with standardised technology independent interfaces	**No**, system does not permit radio applications implementing RATs to be constructed from multiple software components	**Undefined**, although the system is capable of supporting Reconfigurable Protocol Stacks, this functionality is not formally defined	**Yes**, Radio Applications for specific RATs are constructed from individual software components that are assigned to specific predefined layers

2.3.3.2 Using Reconfigurable Protocol Stacks to support any RAT

The majority of the reviewed RRSs utilise the RPS methodology, as it provides a degree of flexibility and scalability which cannot be achieved via traditional monolithic radio application modelling methods. The optimal method of implementing a RPS is the Composable Protocol Stack approach, as it does not limit the radio application to any predefined layering format. To guarantee software component interoperability, a standardised methodology to decompose RATs into their fundamental components must also be provided as part of the RPS specification.

2.4 Summary

This chapter has provided detailed definitions and explanations for the terms used to describe different aspects of research relating to RRSs, and has examined each of the major RRS research and standardisation initiatives with a focus on their requirements, architectures, and functionality. An analysis of the capabilities of each major RRS resulted in the development of a set of unified functional requirements for the Operating Environment, Reconfiguration Management system, and Radio Applications of a new Converged RRS Architecture which contains all of the functionality of the reviewed RRSs. The following chapter examines how these functional requirements are implemented in the Converged RRS Architecture, and describes its logical components, standardised interfaces, and RRS management operations.

Chapter 3

The Converged Reconfigurable Radio System Architecture

The Converged Reconfigurable Radio System Architecture unifies the functionality found in all major researchers' and standards bodies' RRS systems into a single complete logical architecture, which provides an operational environment, a management system, and which implements RATs using the Reconfigurable Protocol Stack approach.

The previous chapter defined a set of functional requirements for the CRRSA, following an examination and comparison of all major RRS systems. The purpose of this chapter is to describe how those functional requirements are encapsulated and implemented within the CRRSA, using a model-driven development process and the engineering design practices outlined at the end of Chapter 1.

The following section provides an overview of the CRRSA, its logical components, and its interfaces. As each of these is examined, references are made to the remaining sections in the chapter where such topics are reviewed in greater detail. Early versions of this work were published in [92, 93].

3.1 Overview of the Converged Reconfigurable Radio System Architecture

The CRRSA is a logical architecture containing abstracted logical components, standardised interfaces, and predefined operating procedures. The architecture was developed using a model-driven development approach, thereby making it technology

independent. During implementation, the developer must carefully choose which technologies to use, so as to ensure that portability and interoperability between the architecture's logical components are maintained and are not compromised. A middleware solution such as CORBA, or the JAVA programming language and virtual machine, are two potential candidates.

In any architecture or system there is always a trade-off between flexibility and scalability, and between interoperability and portability. In the case of the CRRSA both of these sets of criteria are considered vital; however as one of the primary goals of an RRS system is to ensure that it is capable of being upgraded to support any future RAT technology, there are several instances where flexibility and scalability are considered to be of a higher priority. This principle is examined in greater detail in the appropriate sections in this chapter.

As previously mentioned, the CRRSA is composed of logical components, each of which has a set of predefined responsibilities relating to one or more of the functional requirements established in the previous chapter. In addition to the broad description of each component's responsibilities, the CRRSA also provides details regarding how each component must perform its allocated tasks, and the format and content of messages which are passed via the interfaces between components.

Each logical component is assigned to one of three domains, indicating the scope of each component's responsibilities. Logical components assigned to the core network domain perform operations designed to manage and control all reconfiguration processes throughout the architecture; components assigned to the access network are located in each network element in the RAN (typically the base station) and are responsible for managing and controlling all reconfiguration operations involving connections between subscribers' mobile devices and the RAN, and finally all logical components located in the subscriber domain manage and control the reconfiguration functions within each mobile device.

The CRRSA's core network domain, its logical components, and all interfaces are defined in Figure 3.1. The architecture abstracts all core networks functions which are outside the scope of the CRSSA's responsibilities and represents them as a single block called the Abstracted Core Network. Similarly in the other two domains, the upper layers of the protocol stack are represented by a logical component referred to as the Network Layer, which includes network and transport layer protocols, applications and services, and backhaul mechanisms linking the RAN and the core network.

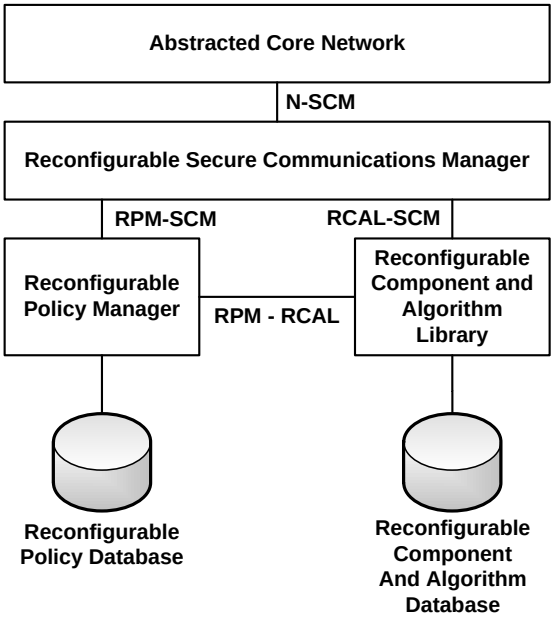


Figure 3.1: The Converged Reconfigurable Radio System Architecture's Core Network Domain

The primary purpose of the logical components within the CRRSA's core domain is to provide network operators with the ability to manage and control all reconfiguration operations in the architecture through the use of a policy-based management system and a software manager to control downloads and to perform updates on all components within the architecture. This functionality is encapsulated in the Reconfigurable Policy Manager (RPM), defined in Section 3.5, and the Reconfigurable Component and Algorithm Library (RCAL), defined in Section 3.4. Transmission of RRS management messages between the Core Network and the Access Network domains, and between the Core Network and Subscriber domains, is managed by the Reconfigurable Secure Communications Manager (SCM), defined in Section 3.6. The SCM is responsible for ensuring secure reliable communications between the domains, and can be upgraded via a novel mechanism called a Reconfigurable Algorithm, defined in Section 3.2.

Figure 3.2 specifies the CRRA's Subscriber and Access Network domains. Both domains possess the same logical components, interfaces and general architecture. The reason for this is that despite the differences in implementation requirements and technologies, the reconfiguration functions performed in the subscriber and access network domains are virtually identical. The differences in radio resource

requirements, spectrum allocations and RATs implemented are all managed though the use of separate policies and software components for each domain.

Instances of the SCM, RPM, and RCAL logical components are also present in both the subscriber and access network domains, which act as counterparts to the corresponding instances in the core network.

The Reconfigurable Radio Resource Manager (RRRM) provides the CRRSA's cognitive functionality, and is responsible for deciding when to perform, and then to manage, reconfiguration operations. The RRRM, like the SCM, also utilises a Reconfigurable Algorithm, which allows a network operator to upgrade or adjust the method used to decide when and how to perform reconfigurations.

The RRRM acquires its context information from the policies supplied by the RPM, and from the RATs' status and performance measurements gathered and compiled by the Reconfigurable Monitoring Module (RMM).

After the RRRM reaches a decision, it instructs the Reconfigurable Controller (RC) to either create or terminate a RAT, or to adjust the parameters governing how different logical components in the CRRSA perform their functions. The RC acquires instances of the software components used to create RATs from the RCAL.

The responsibilities of, and operations performed by, the RRRM, RMM, and RC are presented and defined in detail in Section 3.7. Due to its inherent flexibility and scalability, the Composable Stack approach to Reconfigurable Protocol Stacks is used to model and implement RATs in the CRRSA. The software components used to construct each RAT are called Reconfigurable Components, and these are defined in Section 3.3.

The RRRM manages inter-RAT traffic scheduling through the Packet Multiplexer (PMUX), defined in Section 3.8. This logical component provides all RATs with an interface to the Network Layer, and contains a Reconfigurable Algorithm to enable the network operator to adjust the inter-RAT traffic routing algorithm.

The Waveform Multiplexer (WMUX) is responsible for multiplexing the signals supplied by each RAT prior to their transmission over the wireless environment. The WMUX also provides an arbitration function to resolve different RATs' conflicting requests for transmission and/or reception parameters. The WMUX is the final logical component to contain a Reconfigurable Algorithm, thereby providing the means for its functionality to also be adjusted or upgraded as required

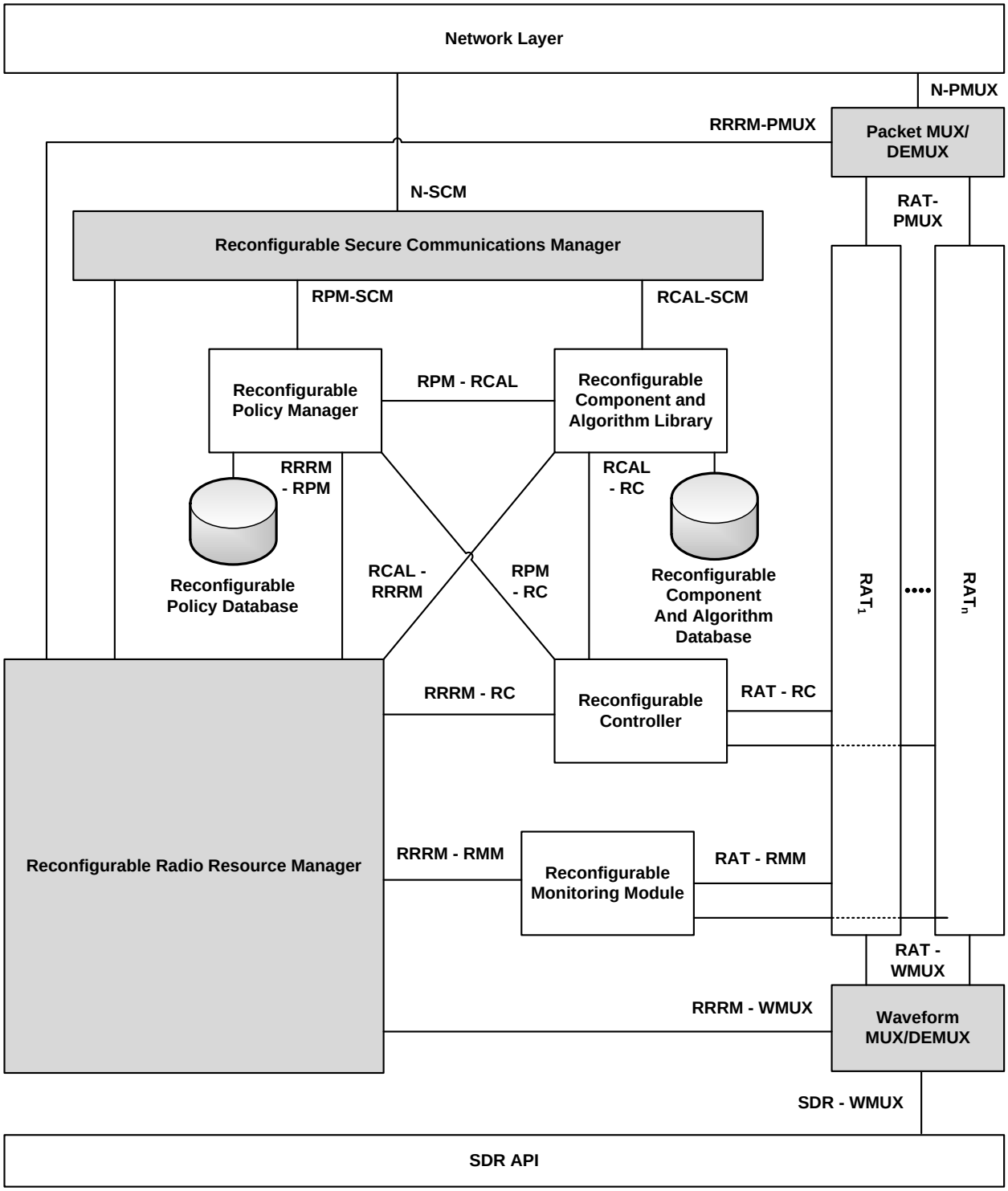


Figure 3.2: The Converged Reconfigurable Radio System Architecture's Subscriber and Access Network Domains

The SDR Application Programming Interface (API), together with the WMUX, provides RATs with a standardised interface to the SDR hardware platform. The SDR API only specifies the services which must be provided to RATs, allowing SDR hardware platform vendors the independence to develop their own hardware systems without compromising the interoperability and portability of CRRSA implementations. The WMUX and SDR API are presented and defined in Section 3.9.

3.2 Reconfigurable Algorithms

Reconfigurable Algorithms are standardised logical components which contain specific implementations of various functions performed by the management systems contained within the CRRSA. Reconfigurable Algorithms enable the CRRSA's management systems to be customised for specific deployment scenarios, or to enable such systems to be upgraded to include new functionality and/or technologies. Figure 3.3 defines the Reconfigurable Algorithm concept.

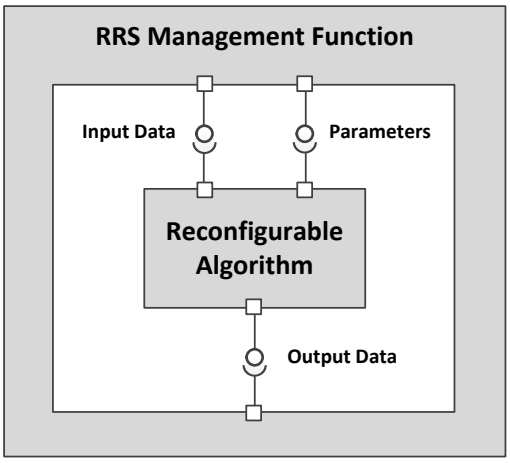


Figure 3.3: The Reconfigurable Algorithm Concept

There are four logical components within the CRRSA which implement Reconfigurable Algorithms; they are the RRRM, the SCM, the PMUX, and the WMUX. Each of these logical components acts as a wrapper for a specific type of Reconfigurable Algorithm, and provides the algorithm with the relevant input data and parameters, and processes and reacts based on the resultant output data.

Each implementation of a specific type of Reconfigurable Algorithm is expected to perform certain predefined tasks; the process and parameters used to complete these are however not specified by the CRRSA. The purpose of this approach is to afford developers the freedom to implement and upgrade each type of algorithm according to their own requirements, thereby enabling the customisation and continual improvement of the functionality of each CRRSA implementation.

3.3 Reconfigurable Components

The primary differentiator between different RATs lies in the air-interface, which provides Medium Access Control (MAC), mobility management, QoS mechanisms, and a variety of other functions to establish and maintain the link between the subscriber and access network domains, as well as signal processing techniques to process and modulate packets in order to ensure that they are successfully transmitted over the wireless environment. Although modern RATs do not always fit neatly into the OSI model's layering scheme, these air-interface functions can be loosely viewed as existing within the Data Link Layer (DLL) and Physical Layer respectively.

The CRRSA implements RATs using a set of logical building blocks called Reconfigurable Components. Each Reconfigurable Component encapsulates a single Physical or DLL atomic function, and provides a standardised set of interfaces to connect Reconfigurable Components to one another, the PMUX, and the WMUX. The implementation of a Reconfigurable Component can be accomplished through the use of inheritance, where an abstract class is used to define the Reconfigurable Component's standardised interfaces and operations, and each subclass contains a different atomic function.

The creation of a RAT requires the RC to request instances of specific Reconfigurable Components from the RCAL, and to connect the Components together in a topology predefined via policies. Once the required Reconfigurable Components have been instantiated and connected, each Component operates continuously and autonomously until it is terminated by the RC. The following subsections provide a set of rules which are used to decompose RATs into their fundamental components, describe and specify the architecture of a Reconfigurable Component, and define the internal operations performed within each Reconfigurable Component and the messages which are passed between them.

3.3.1 Rules for the Decomposition of RATs and development of Reconfigurable Components

The primary motivation for the development and implementation of RRSs is that their functionality can be upgraded through the installation of new software components, and that such components can be reused and ported to different hardware platforms. To ensure that software components can be ported to different platforms, RRSs typically specify the functions to be performed by each component and also standardise the interfaces between them. This approach works well when the functions performed by each software component are predefined; however in the case of Reconfigurable Components the different atomic functions which could be encapsulated are virtually limitless. In order to prevent different developers from producing separate sets of Reconfigurable Components with overlapping atomic functions, thereby subverting the portability of the components, a set of rules which establishes the procedure for RAT decomposition and development of Reconfigurable Components is presented and defined below:

1. Each Reconfigurable Component must contain only a single atomic function.
2. Developers must make every attempt to implement each atomic function in a manner which allows it to be reused in another RAT i.e. if parts of the atomic function can be adapted, input parameters must be provided to enable each RAT to adapt the atomic function to meet its own needs.
3. In the event that a specific atomic function is unique to a particular RAT, such a function must be encapsulated in its own Reconfigurable Component, independent of any reusable generic functions.
4. Each Reconfigurable Component must possess at least one input or one output interface. Reconfigurable Components without any output interfaces are useful for performing RAT status and performance measurements, and passing the results to the RMM. Reconfigurable Components without any input interfaces can be used to provide static values as inputs to other Reconfigurable Components.
5. If a developer is unsure whether to decompose a specific set of atomic functions within a RAT into one or more Reconfigurable Components, the following primary rule provides guidance: If there are three or more Reconfigurable Components containing generic reusable RAT atomic functions which are always interconnected with one another regardless of the implementation, then

the functions of those Reconfigurable Components should be merged into a new Reconfigurable Component. This rule is defined in Figure 3.4.

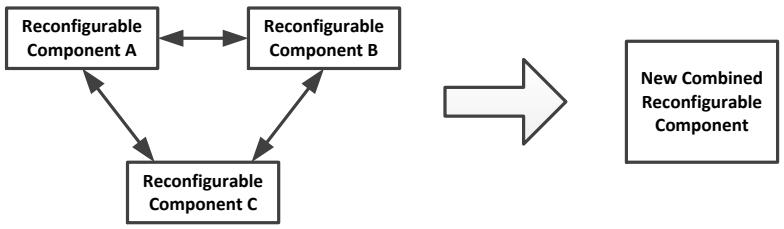


Figure 3.4: Primary Rule for RAT Decomposition into Reconfigurable Components

3.3.2 Overview of the Reconfigurable Component Logical Architecture

The logical architecture of a Reconfigurable Component, defined in Figure 3.5, consists of a set of standardised interfaces to connect it to other logical components, a routing table which contains the list of connections, the atomic function which processes inputs and produces outputs, a set of queues to buffer output values until they are needed, and a system to manage and control all operations.

The Reconfigurable Component logical architecture specifies three types of interfaces, each of which carries a different type of traffic. Bearer interfaces are used to convey standard user data relating to mobile communication services, Control interfaces carry signalling traffic used to manage the connection over the wireless environment, and RRS Management interfaces, provide the RC with the ability to create and terminate each Reconfigurable Component and the RMM with the means to gather context information.

When the RC first constructs a RAT, each Reconfigurable Component is supplied with a list of connections to other Reconfigurable Components, the PMUX, and/or the WMUX. These connections are stored in the Reconfigurable Component's routing table, and are static for the duration of the RAT's existence. Connections

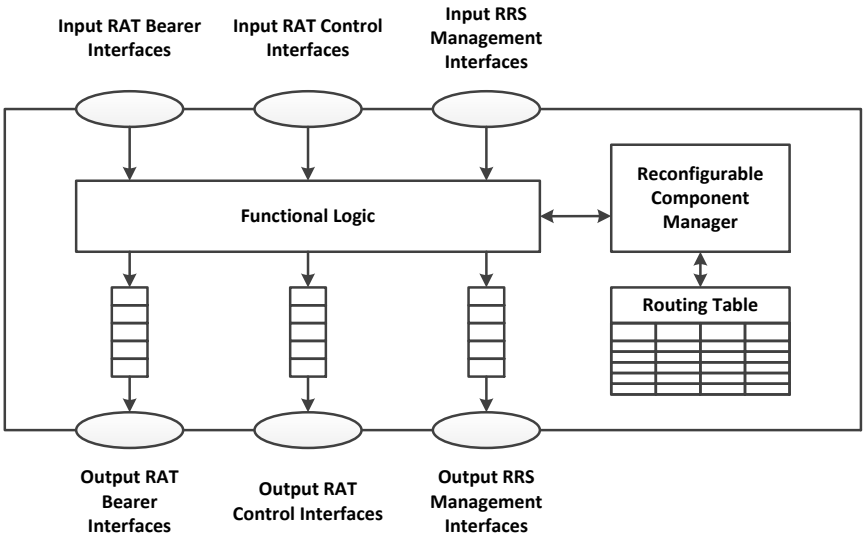


Figure 3.5: Reconfigurable Component Logical Architecture

between Reconfigurable Components are established using a client-server relationship, whereby each Reconfigurable Component requests input values from another Reconfigurable Component.

As the runtime of each Reconfigurable Component depends on its encapsulated function, each Reconfigurable Component may produce a result at a different rate to others, and this therefore requires each Reconfigurable Component to buffer its results in an output queue until they are requested by another Reconfigurable Component. To manage the flow of messages throughout a RAT, each Reconfigurable Component only requests and processes new inputs once its output queue(s) have dropped below a specified threshold. This ensures that some Reconfigurable Components do not produce results faster than others can process them. During the development of each Reconfigurable Component, input and output specifications must be determined and saved in the routing table. These specifications provide a unique ID for each input and output, a name to identify the input or output, the type of interface, and the nature of the input or output value, i.e. the data type. These specifications are combined with the connection information received when the Reconfigurable Component is created, and are used to identify and differentiate inputs and outputs during inter-Reconfigurable Component communications.

In addition to the specifications listed above, input connections must also be classified as either Independent or Dependent. Independent inputs are always requested by a Reconfigurable Component prior to processing them using the encapsulated atomic function. Dependent inputs on the other hand, are only requested subject to the values of already acquired Independent inputs or other Dependent inputs. The logic to determine whether a Dependent input should be requested forms part of the atomic function and is therefore implementation specific. The benefit of the Dependent input mechanism is that it provides the means to dynamically adapt the functionality provided by each Reconfigurable Component.

Finally, the Reconfigurable Component also provides a set of standardised functions which can be used by the Component's encapsulated atomic function to send context information to the RMM, request measurements of the wireless environment from the SDR, and request changes to the radio resources allocated for transmission and reception by the SDR.

3.3.3 Reconfigurable Component Operations and Communication

Although each developer is free to implement each atomic function using whatever methodology is most appropriate, the remainder of the operations performed by the Reconfigurable Radio Component are standardised to ensure interoperability.

Figure 3.6 provides a detailed description of the standard operations performed by each Reconfigurable Component, and Figure 3.7 defines the communication process between a set of Reconfigurable Components whilst they are performing their standard operations. As different Reconfigurable Components produce results at different rates, a Reconfigurable Component's operations are divided into two sets of tasks which must be performed simultaneously. The Input and Processing task set involves checking that all output queue lengths have fallen below a predefined threshold, requesting and receiving all independent and dependent inputs, processing the inputs using each Reconfigurable Component's atomic function, and finally saving the results to specific output queues. The Output Distribution task set is responsible for responding to requests for output values from other Reconfigurable Components.

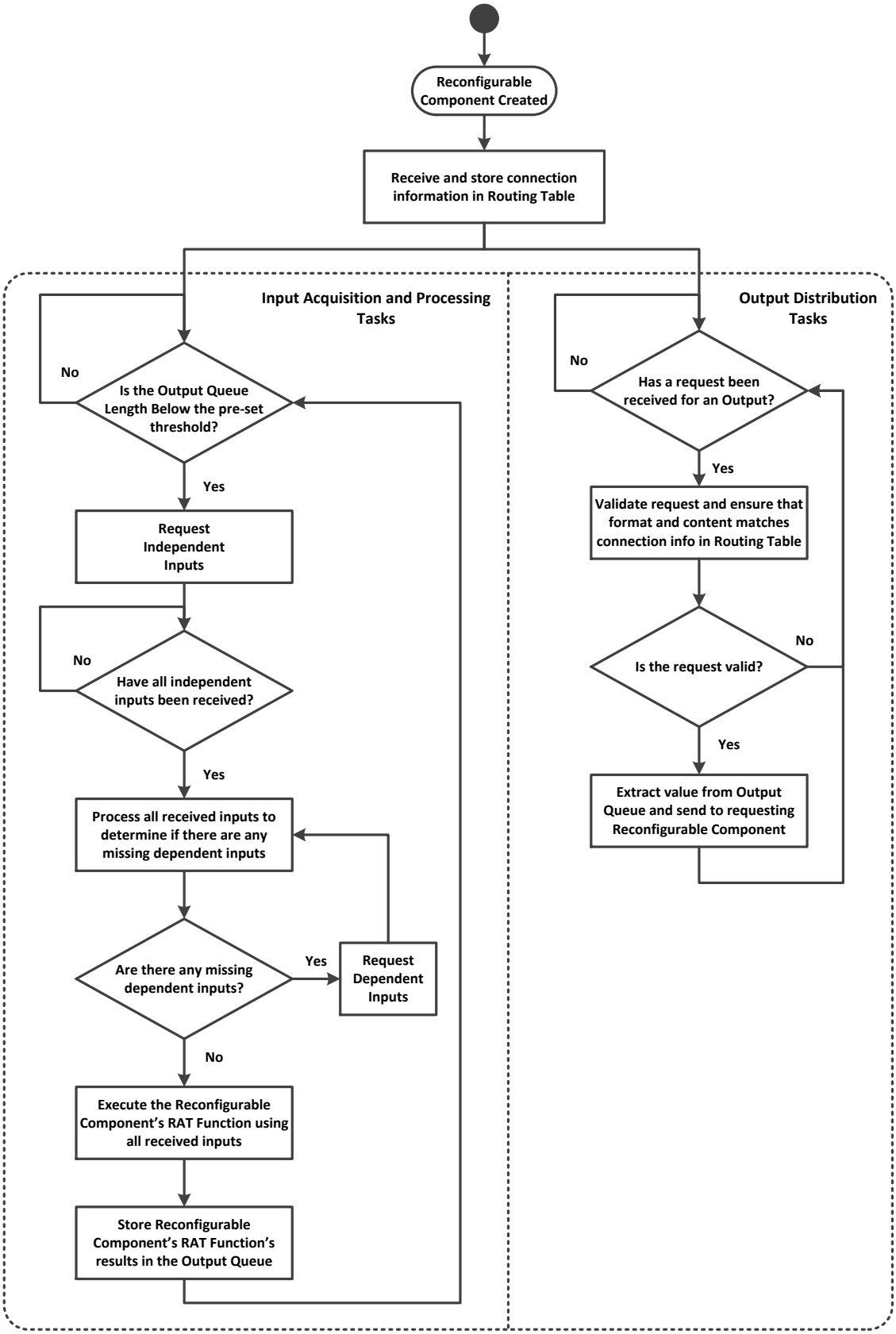


Figure 3.6: The internal operations of a Reconfigurable Component

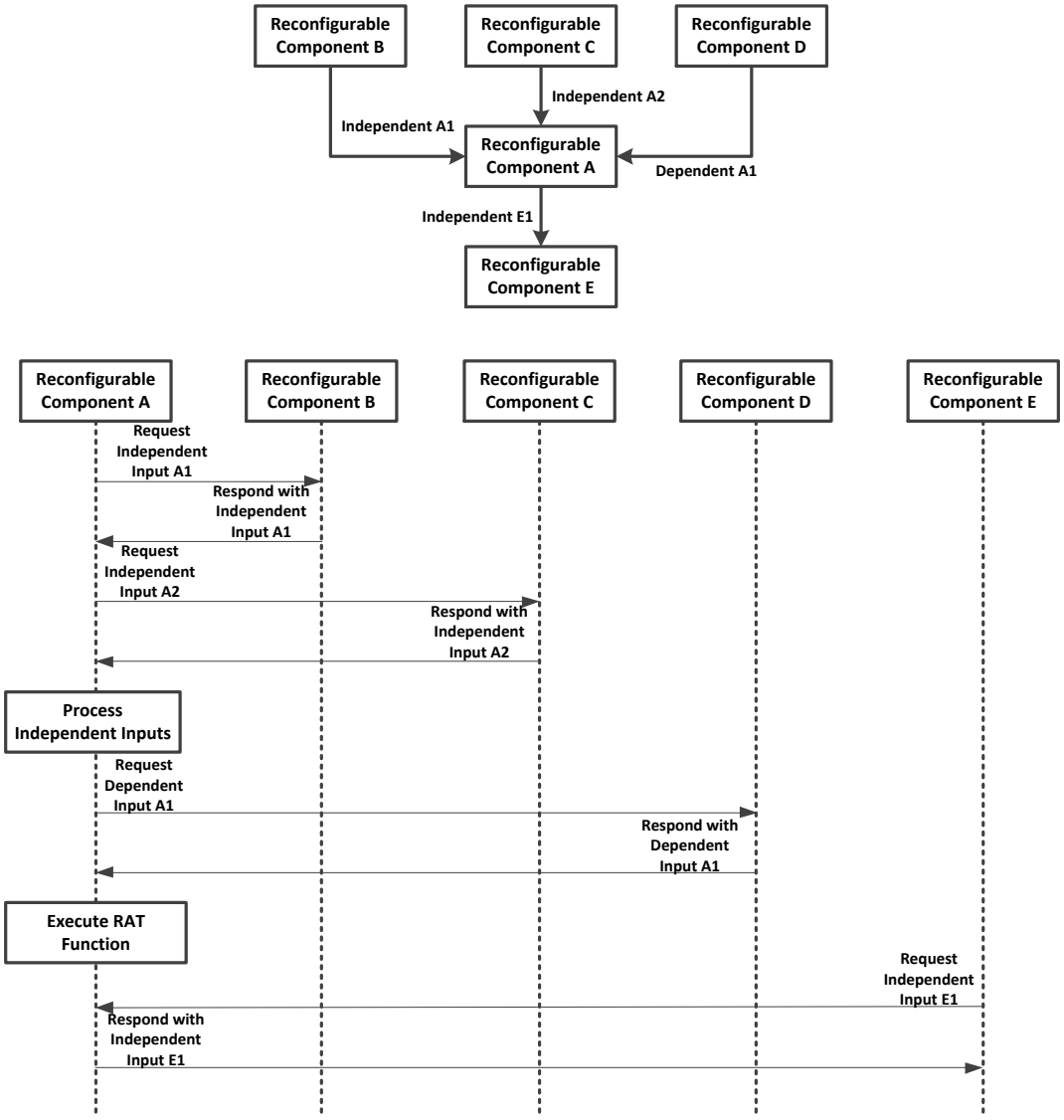


Figure 3.7: Communications between Reconfigurable Components

3.4 Managing Reconfigurable Algorithms and Components

The RCAL is the logical component within the CRRSA responsible for downloading, validating, storing, and instantiating all Reconfigurable Components and Algorithms across all domains. The RCAL in the Core Network domain provides the network operator with the ability to validate and distribute new Reconfigurable Components and Algorithms to all network elements in the Subscriber and Access Network domains.

The RCALs in the Subscriber and Access Network domain periodically send requests for new software components to the Core Network RCAL, along with a list of the software components which each RCAL already possesses. The Core Network RCAL determines whether there are any new software components to be sent to the requesting RCAL, and responds with any components required. All mobile devices in the subscriber domain must possess preloaded Reconfigurable Components and Algorithms so they are able to construct a RAT, and to establish a link to the Core Network RCAL.

As the functionality implemented in Reconfigurable Components and Algorithms is practically limitless, the validation process performed by each RCAL is of necessity limited to ensuring that new software correctly implements all internal operations and interfaces, and that such software can be compiled and executed.

After the validation process is completed, the RCAL sends the RPM a list of the validated Reconfigurable Components and Algorithms. The RPM uses this information to validate its policies, and to determine whether the implementation of each policy is feasible.

The RCAL is also responsible for providing instances of Reconfigurable Components to the RC and Reconfigurable Algorithms to the RRRM. The RC uses Reconfigurable Component instances to construct new RATs, whilst the RRRM passes Reconfigurable Algorithm instances with parameters to the SCM, PMUX, and WMUX.

As the Core Network domain does not contain a RRRM, the RCAL must provide an instance of the appropriate Reconfigurable Communications Algorithm to the SCM, based on a request from the RPM.

Figure 3.8 provides a detailed description of the internal operations performed by the Core Network RCAL, as well as by the Subscriber and Access Network RCALs.

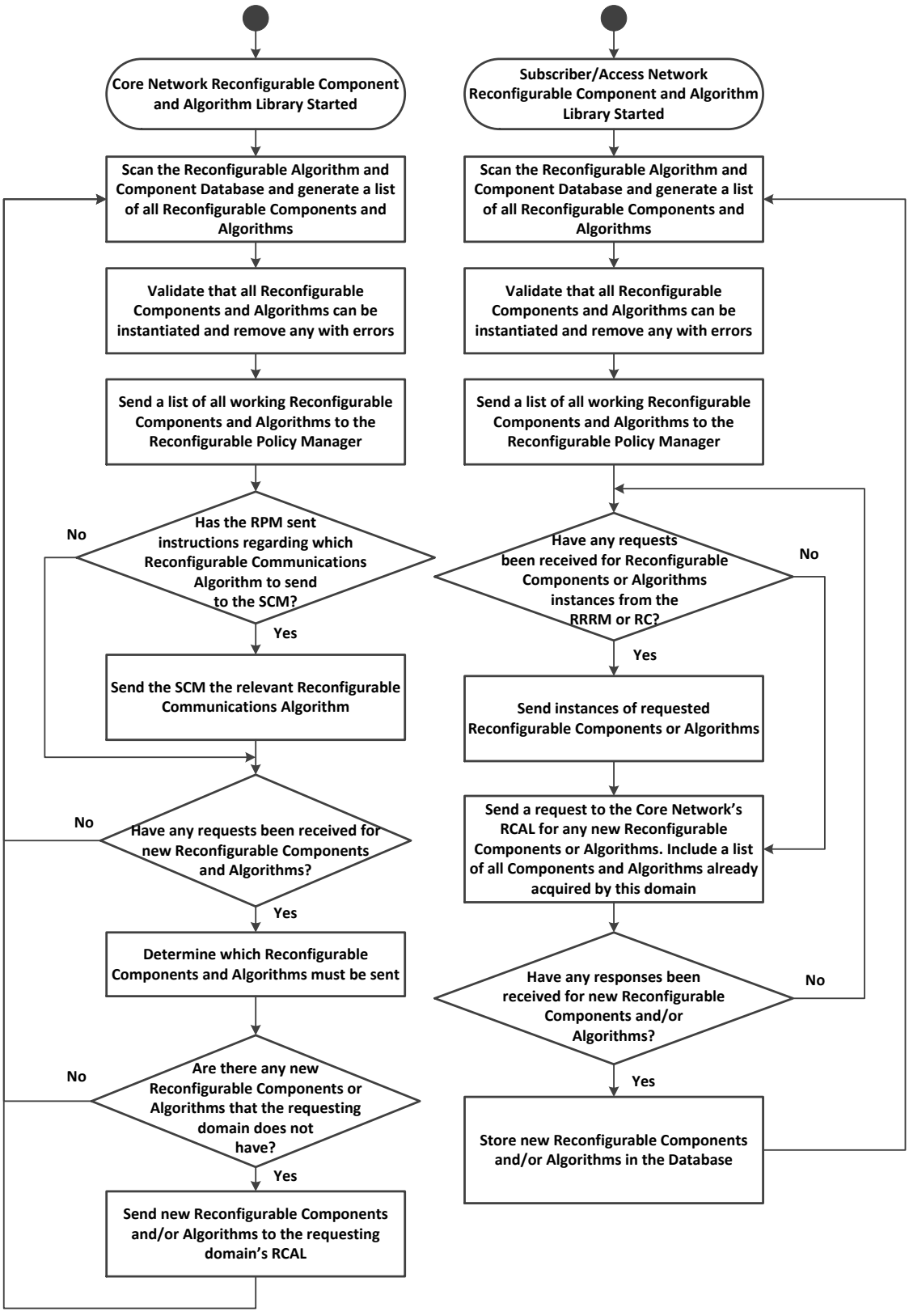


Figure 3.8: The internal operations of the RCAL in the Core Network, and Subscriber and Access Network domains

3.5 Policy-based Management

Before any reconfiguration decision can be taken the RRRM requires a set of predefined objectives and constraints, and a list of the available RATs which may be implemented. Policies are the means by which this information is provided to the RRRM by the network operator. The use of policies is beneficial as it allows reconfiguration decisions to be taken and executed autonomously.

There are two types of policies in the CRRSA, namely Network Equipment policies and RAT policies. The distribution and validation of these policies is carried out by the RPM. The following subsections define each set of these policies.

3.5.1 Network Equipment Policies

Network Equipment policies are specific to each network element within the subscriber and access network domains, and only one Network Equipment policy may be active in each network element at any time. These policies contain a list of the RATs which each network element is permitted to implement, and the names of and default parameters for the Reconfigurable Algorithms which must be used. This provides the network operator with the ability to specify different RATs and operational reconfiguration parameters for each network element in the subscriber and access network domains. Examples where this would be beneficial include: limiting a subscriber to specific RATs based on his or her contract, or specifying different spectrum allocations for different network elements within the RAN.

Reconfigurable Algorithms control how the RRS management system communicates across domains, how traffic is scheduled between RATs, how reconfiguration decisions are made, and how radio resources and transmission parameters are managed. Thus the radio resources and spectrum allocations for all network elements in the subscriber and access network domains can be optimised by adjusting which parameters and Reconfigurable Algorithms are implemented.

Although Network Equipment policies specify the default parameters and Reconfigurable Components to be implemented by each network element, these can be modified by the RRRM if the Reconfigurable Decision Making Algorithm as implemented possesses the ability to do so. This approach enables the RRRM to perform dynamic radio resource and spectrum allocation.

Aside from specifying which RATs a network element is permitted to implement,

the Network Equipment policy also stipulates which of those RATs should be implemented when the network element is turned on. These RATs, referred to as default RATs, provide CPC functionality to the CRRSA, and enable mobile devices to establish an initial connection to a network element in the RAN. After a connection has been established, the mobile terminal may request or be instructed to implement a different RAT.

As default RATs are specified in Network Equipment policies, this provides a network operator with the ability to change them as new RATs are developed and older legacy mobile communications systems are phased out.

3.5.2 Radio Access Technology Policies

Each Radio Access Technology Policy contains performance metrics for a specific RAT, and a list of the Reconfigurable Components with connections which is used to implement the RAT. Like Network Equipment policies, RAT policies are specific to each network element in the subscriber and access network domains. This means that different RAT implementations can be provided for mobile devices and network elements in the RAN. This functionality is necessary, as most RATs have separate (and different) implementations for their Uplink and Downlink transmissions.

The RAT performance metrics included in RAT policies provide the RRRM with the information it needs in order to determine which RAT it should implement based on the context information it receives from the RMM. The CRRSA does not specify which performance metrics should be included in RAT policies as these may change when new RATs and technologies are introduced, however possible metrics include: the transmission bandwidth, the carrier frequency, the average transmission power, the maximum data throughput, the optimal spectral efficiency, and the duplexing scheme to be employed.

The RAT Policy also provides the RC with the information it needs to construct its RAT. This information includes the list of Reconfigurable Components used to create the RAT, the connections between all Reconfigurable Components, the connections to the PMUX and WMUX, and the input and output specifications for each Reconfigurable Component. To ensure that there are no errors, these input and output specifications must match those provided by a Reconfigurable Component or the RAT policy is considered invalid.

3.5.3 The Reconfigurable Policy Manager

The Network Equipment and RAT Policies are managed and distributed by the RPM. The primary responsibility of the RPM is to distribute specific policies to each network element, validate any received policies, and supply them to the various logical components in the CRRSA as required.

The policy validation process is performed on all policies after they have been discovered in an RPM database or received from the Core Network domain, and is executed in two stages. The first stage requires the list of available Reconfigurable Components from the RCAL, which it in turn compares to the reconfigurable components listed in each RAT policy. In order for a RAT policy to be considered valid, all listed Reconfigurable Components must be available, the input and output specifications specified in the policy must match those of the Reconfigurable Components, there must be no interfaces without a connection to another logical component, and the connections to the PMUX and WMUX must be specified.

The second stage of the validation process requires the list of available Reconfigurable Algorithms from the RCAL, which it compares to those listed in the active Network Equipment policy. A Network Equipment policy is considered valid if all the Reconfigurable Algorithms listed in the policy are available for implementation, and if each of the RATs which are permitted to be implemented by the network element have valid corresponding RAT policies. In the event that a Network Equipment policy is declared invalid a new one must be sourced from the Core Network. Should the policy in the Core Network itself be invalid, the Network Operator must replace the invalid policy in the Core Network's policy database with a new one.

The Core Network RPM provides the network operator with the ability to distribute specific policies to all network elements in the subscriber and access network domains. The RPMs in the subscriber and access network domains periodically submit requests for new policies to the Core Network RPM. The Core Network RPM evaluates these requests and responds with new policies if they are available and applicable. All mobile devices must be preloaded with the required Network Equipment and RAT policies, to enable them to establish a link to the Core Network RPM when they are turned on for the first time.

As was previously mentioned, the RPM is also responsible for providing the RRRM with the active Network Equipment Policy, and the RC with any requested RAT policies. Whenever the RPM supplies the RRRM with a new Network Equipment

policy, the RRRM reconfigures all logical components in its domain according to the default Reconfigurable Algorithms, default parameters, and default RATs listed in the policy.

As the Core Network domain does not contain an RRRM, the RPM must examine all available Network Equipment policies to determine which Reconfigurable Communications Algorithm is capable of supporting communications between the Core Network domain and the Subscriber and Access Network domains. The name of this Reconfigurable Algorithm, and its default parameters, are sent to the RCAL.

Figure 3.9 provides a detailed description of the internal operations performed by the Core Network RPM and by the Subscriber and Access Network RPMs.

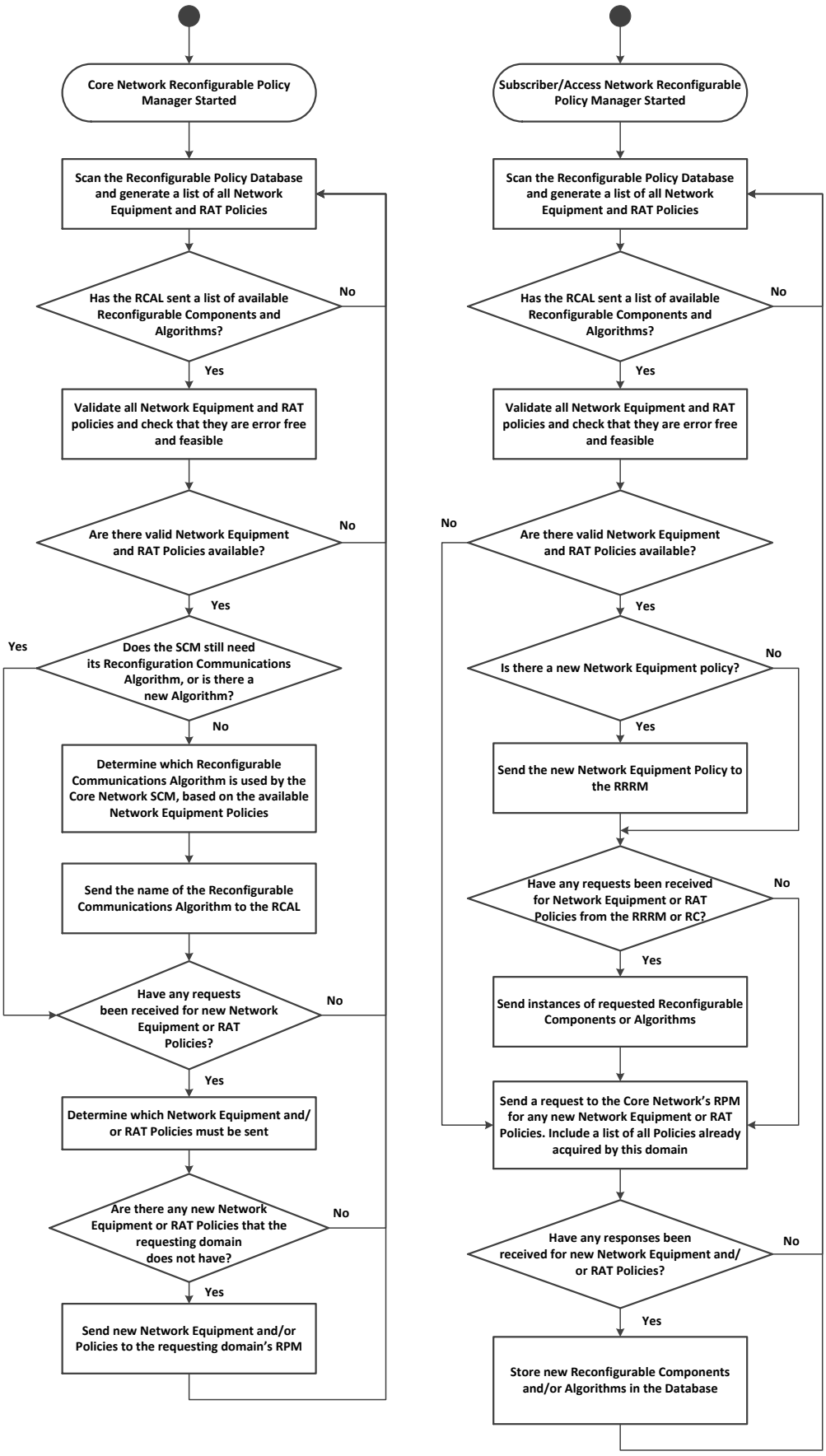


Figure 3.9: The internal operations of the RPM in the Core Network, and Subscriber and Access Network domains

3.6 Securing the Converged Reconfigurable Radio System

The purpose of the SCM is to ensure secure reliable communications between the CRRSA's management systems in each of the domains. The SCM enables the RPMs and RCALs in the Subscriber and Access Network domains to obtain updated policies, Reconfigurable Components, and Reconfigurable Algorithms from the Core Network. Communication between the RRRMs in the Subscriber and Access Network domains is also facilitated by the SCM.

The communications protocol used by the SCM is defined and managed by the current Reconfigurable Communications Algorithm. The SCM can be considered to be a wrapper to the Reconfigurable Communications Algorithm, as it provides the interfaces used to exchange messages with the other logical components, and it translates messages between the format used by the SCM's interfaces and the format which can be processed by the Algorithm.

Instances of the Reconfigurable Communications Algorithm, and the Algorithm's parameters, are supplied by the RRRM in the Subscriber and Access Network domains, and by the RCAL in the Core Network domain.

The Reconfigurable Communications Algorithm is responsible for providing security mechanisms to prevent unauthorised interception or access to transmitted messages, and for providing error control mechanisms to ensure that messages successfully reach the recipient domain.

Although developers are free to implement any communications protocol they choose within a Reconfigurable Communications Algorithm, the standardised interfaces between the SCM and the encapsulated Reconfigurable Communications Algorithm require that each implementation support a specific set of functions. Two of these functions involve processing messages received from logical components in the local CRRSA domain (i.e. the RRRM, the RPM, and the RCAL), and packets received from the Network Layer. The last two functions require the developer to provide processes to deal with transmission and receiving timers expiring. Timers are used by the SCM to detect whether the transmission of one packet in a series of packets has failed, and whether the lost packet needs to be retransmitted.

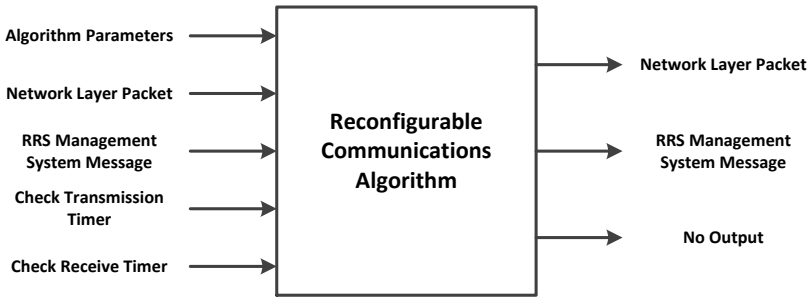


Figure 3.10: The Reconfigurable Communications Algorithm

Figure 3.10 specifies Reconfigurable Communications Algorithm and its Input Data, Algorithm Parameters, and Output Data interfaces. Figure 3.11 provides a detailed description of the internal operations performed by the SCM, and also demonstrates how the functions implemented by the Reconfigurable Communications Algorithm are utilised by the SCM.

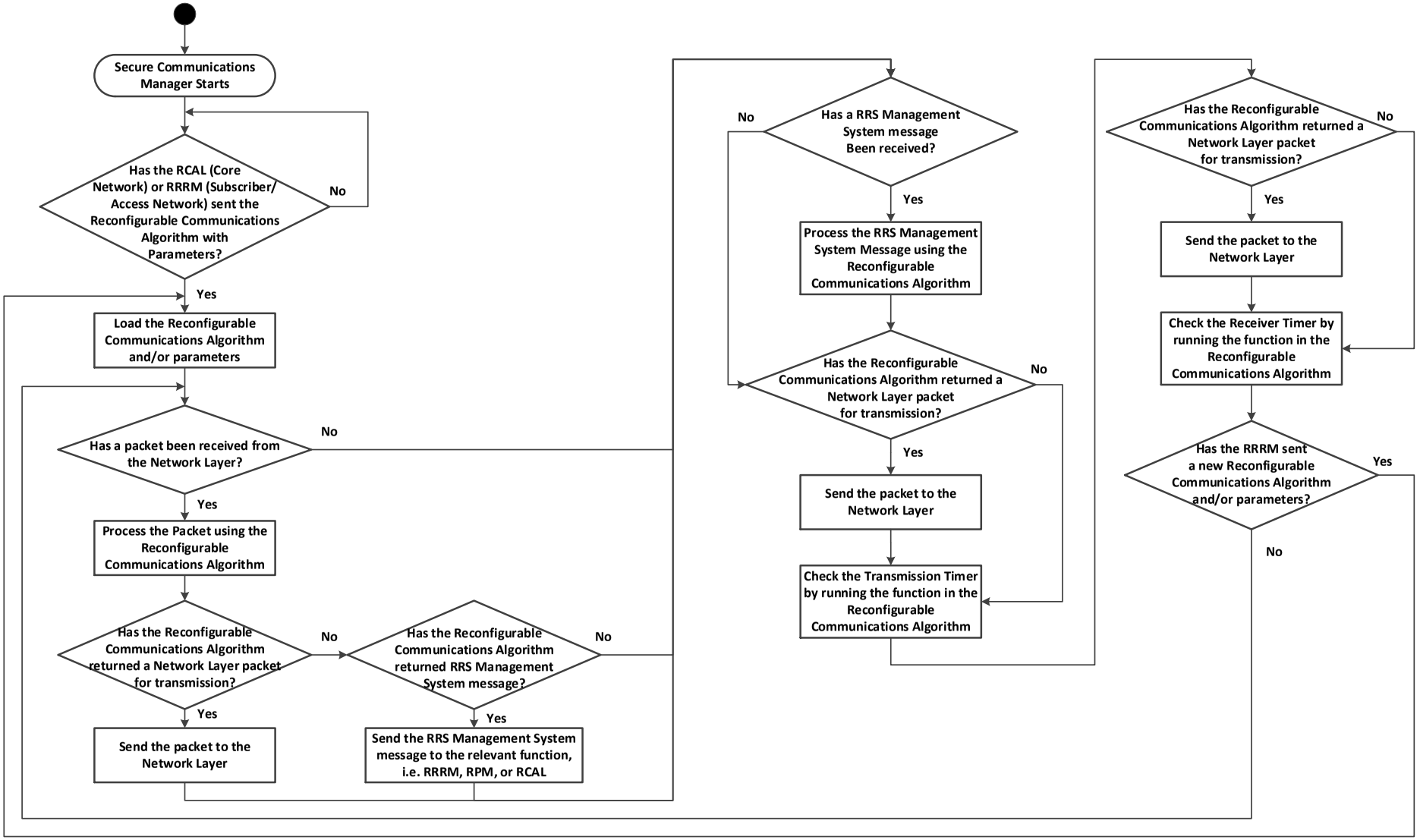


Figure 3.11: The internal operations of the SCM

3.7 Controlling and Performing Reconfigurations

The RRRM is responsible for managing and controlling all reconfiguration operations within the CRRSA. In Cognitive Radio terms, the RRRM is the Cognitive Engine of the architecture. The RRRM's reconfiguration decision making process is implemented in the Reconfigurable Decision Making Algorithm, thereby enabling this functionality to be adapted or upgraded to suit the network operator's requirements or a specific deployment scenario, or in order to incorporate new technologies.

The RRRM, like the SCM, acts as a wrapper for the Reconfigurable Decision Making Algorithm, by proving interfaces and mechanisms for gathering context information, negotiating potential reconfiguration decisions with other RRRMs, and implementing finalised reconfiguration decisions. Figure 3.12 specifies the Reconfigurable Decision Making Algorithm along with its Input Data, Algorithm Parameters, and Output Data Interfaces.

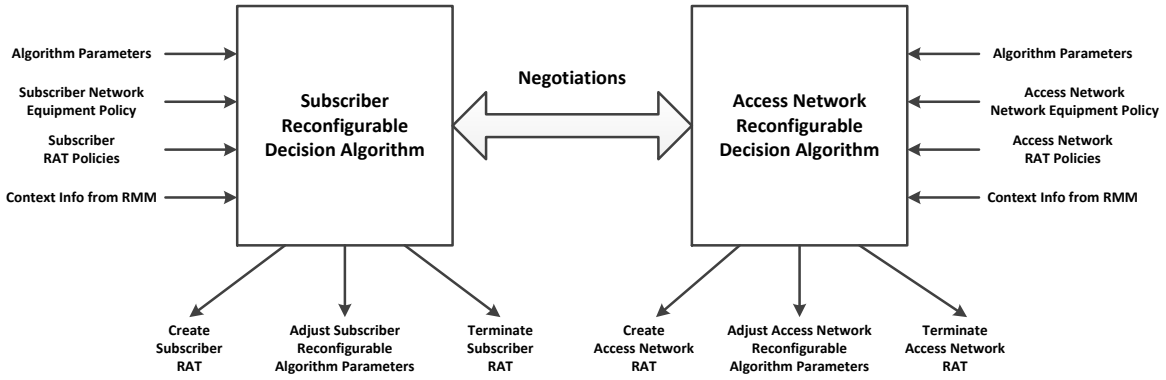


Figure 3.12: The Reconfigurable Decision Algorithm and the Reconfiguration Decision Process

The Reconfigurable Decision Making Algorithm begins its operations by examining the active Network Equipment policy to determine which Reconfigurable Algorithms and default RATs need to be implemented. The Algorithm then uses standardised functions provided by the RRRM to send the relevant Reconfigurable Algorithm instances to the SCM, PMUX and WMUX, and to send the RC instructions to create the default RATs.

After these initial operations have been completed, the Reconfigurable Decision Making Algorithm enters a standardised routine of gathering context information, making reconfiguration decisions, and implementing those decisions.

The following subsections specify in detail the initial and routine reconfiguration operations performed by the RRRM, and define the roles and internal operations of the RC and the RRM when creating and terminating RATs.

3.7.1 Initial Operations and New Network Equipment Policies

When a RRRM instance is created within the Subscriber or Access Network domain, or whenever a new Network Equipment policy is sent from the Core Network, the RRRM's first task is to reconfigure all logical components in the local domain according to the current or new Network policy. Once the RPM has sent this policy to the RRRM, the RRRM examines the specification and requests and then implements an instance of the relevant Reconfigurable Decision Making Algorithm from the RCAL. The RRRM then provides the active Network Equipment policy to the Reconfigurable Decision Making Algorithm.

The Reconfigurable Decision Making Algorithm must first determine which Reconfigurable Algorithms and default parameters to utilise in the SCM, PMUX, and WMUX, obtain instances of these Algorithms from the RCAL, and then send the instances and parameters to their relevant logical components. After this the Reconfigurable Decision Making Algorithm instructs the RC to terminate any existing RATs and then to create all the default RATs specified in the Network Equipment policy. Finally the Reconfigurable Decision Making Algorithm must send the RMM a list of Reconfigurable Components in each new RAT.

Figure 3.13 defines these initial operations in detail, and Figure 3.14 specifies the communications between the logical components involved in this process.

3.7.2 Reconfiguration Decision Making Operations

After the RRRM's initial boot-up operations, or after a new Network Equipment policy has been introduced, the RRRM enters a new state whereby it continuously seeks to determine whether a reconfiguration needs to be performed, and then implements the reconfiguration.

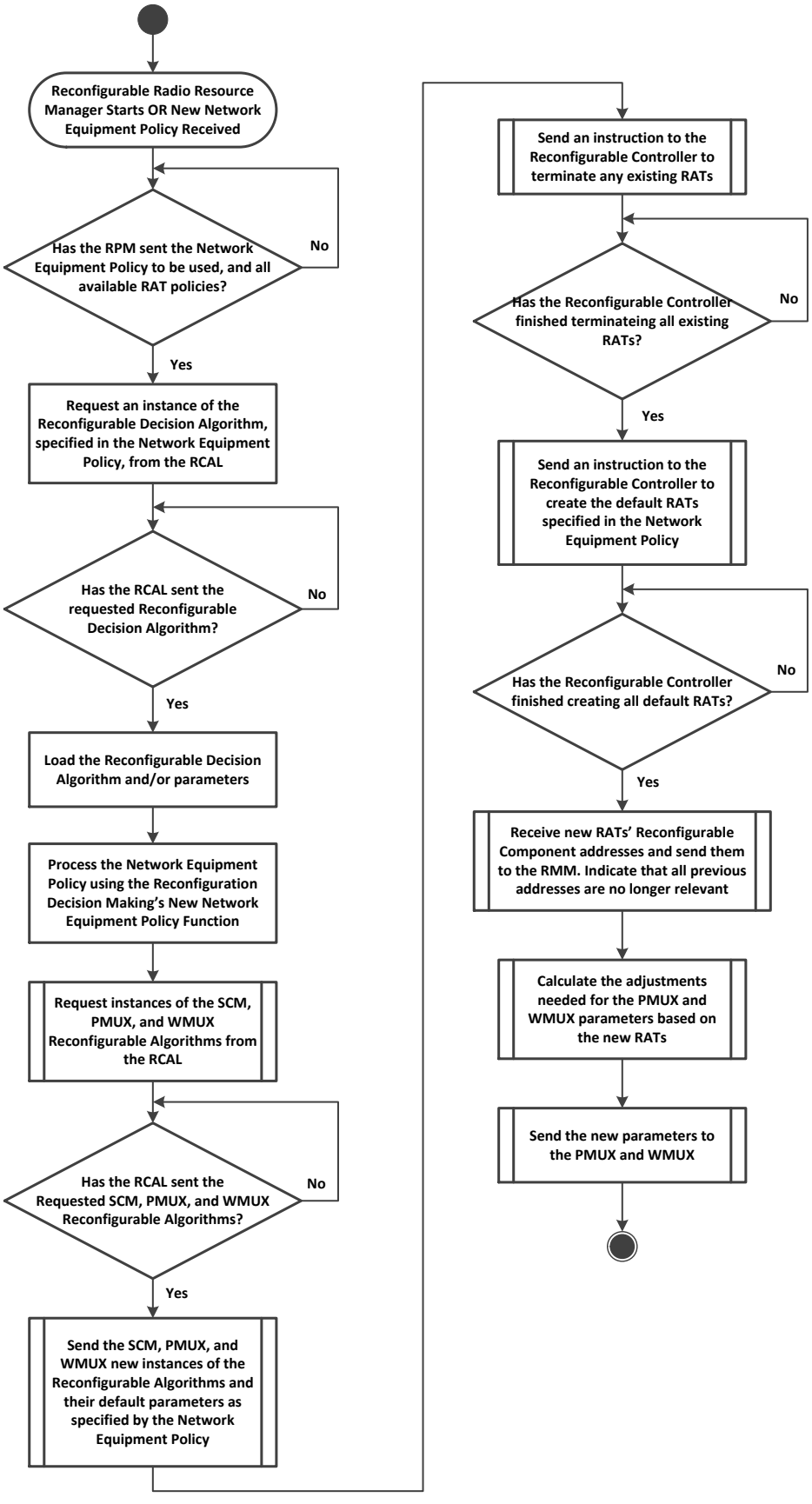


Figure 3.13: The internal operations of an RRRM immediately after initial boot-up or the introduction of a new Network Equipment Policy

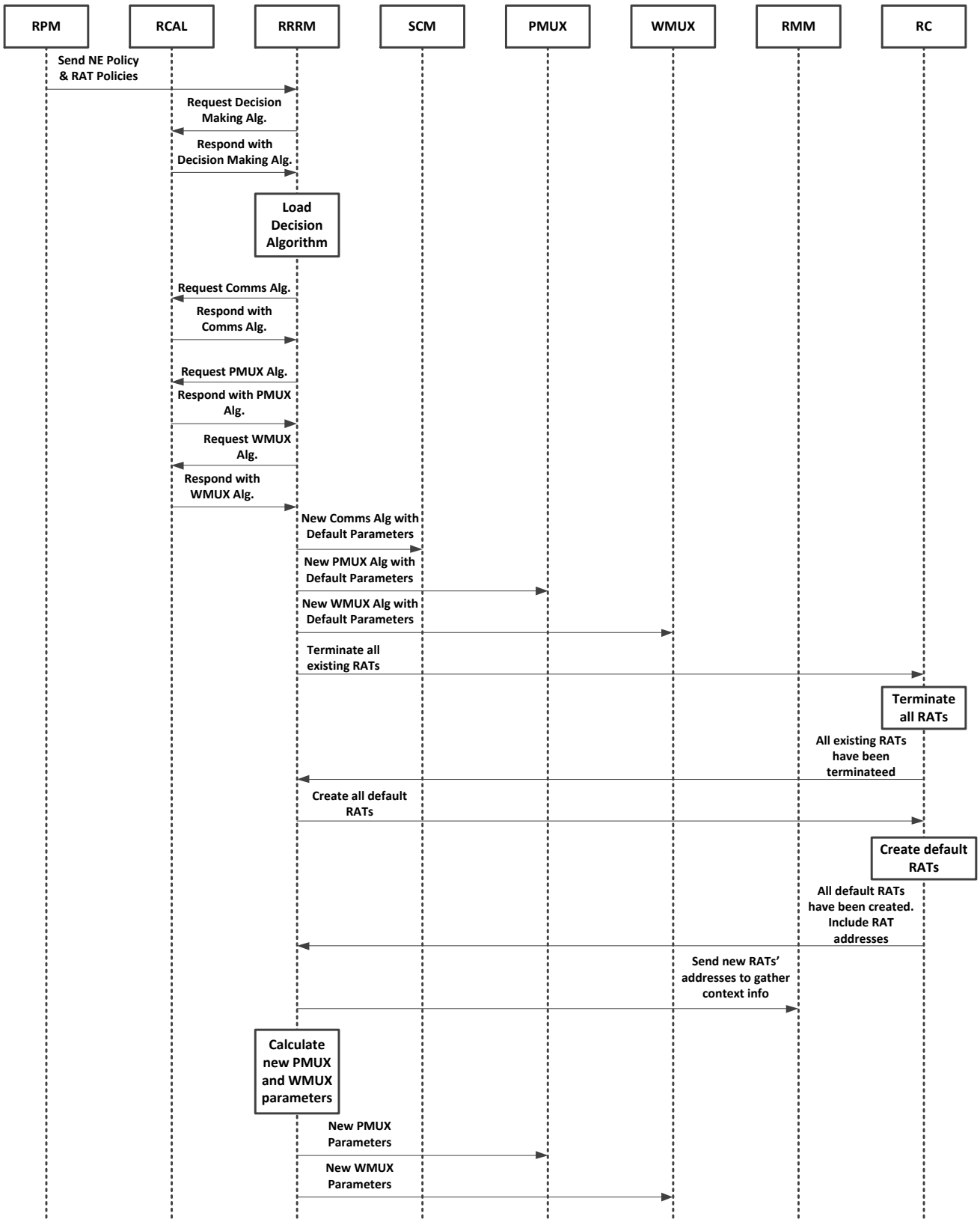


Figure 3.14: Communications between the RRRM and other logical components during initial boot-up or the introduction of a new Network Equipment policy

Context information is supplied to the Reconfigurable Decision Making Algorithm by the RPM and the RMM. The RPM provides the active Network Equipment policy to the Algorithm, which enables it to determine which RATs may be implemented, and it also provides the permitted RATs' policies, which allows the Algorithm to evaluate the performance metrics of each available RAT. The RMM gathers RAT status and performance measurements, and periodically presents these in a compiled report to the Algorithm.

Once the Reconfigurable Decision Making Algorithm has received all the relevant context information, it must determine whether a reconfiguration of the network element's functionality is necessary. If the Algorithm decides to perform a reconfiguration, it must first obtain permission from the corresponding RRRM in the opposing domain, i.e. if a subscriber mobile device decides to perform a reconfiguration, it must first obtain permission from the base station it is connected to, and vice versa.

The purpose of the negotiation phase of the reconfiguration process is to afford the network elements the opportunity to veto the reconfiguration request if it should prove to be impossible or if it would have a negative impact on performance. As each network element uses a separate Network Equipment policy, it is possible to implement different Reconfigurable Decision Making Algorithms in the Subscriber and Access Network Domains. This would provide the network elements in the Access Network domain with the authority to insist on reconfiguration operations being performed, whilst enabling the network elements in the Subscriber domain to make requests which can be either accepted or rejected by the RAN.

After a successful response to a reconfiguration request has been received, the Reconfigurable Decision Making Algorithm can issue one of three separate reconfiguration commands, using standardised functions provided by the RRRM. These commands are: create a new RAT, terminate an existing RAT, or change the Reconfigurable Algorithm and/or parameters in the SCM, PMUX, or WMUX. In the event that an RRRM wishes to swap one RAT for another, a new RAT must first be created, the connections to the old RAT must be transferred to the new RAT, and only then can the old RAT be terminated.

Figure 3.15 provides a detailed description of the RRRM's reconfiguration decision making operations described above.

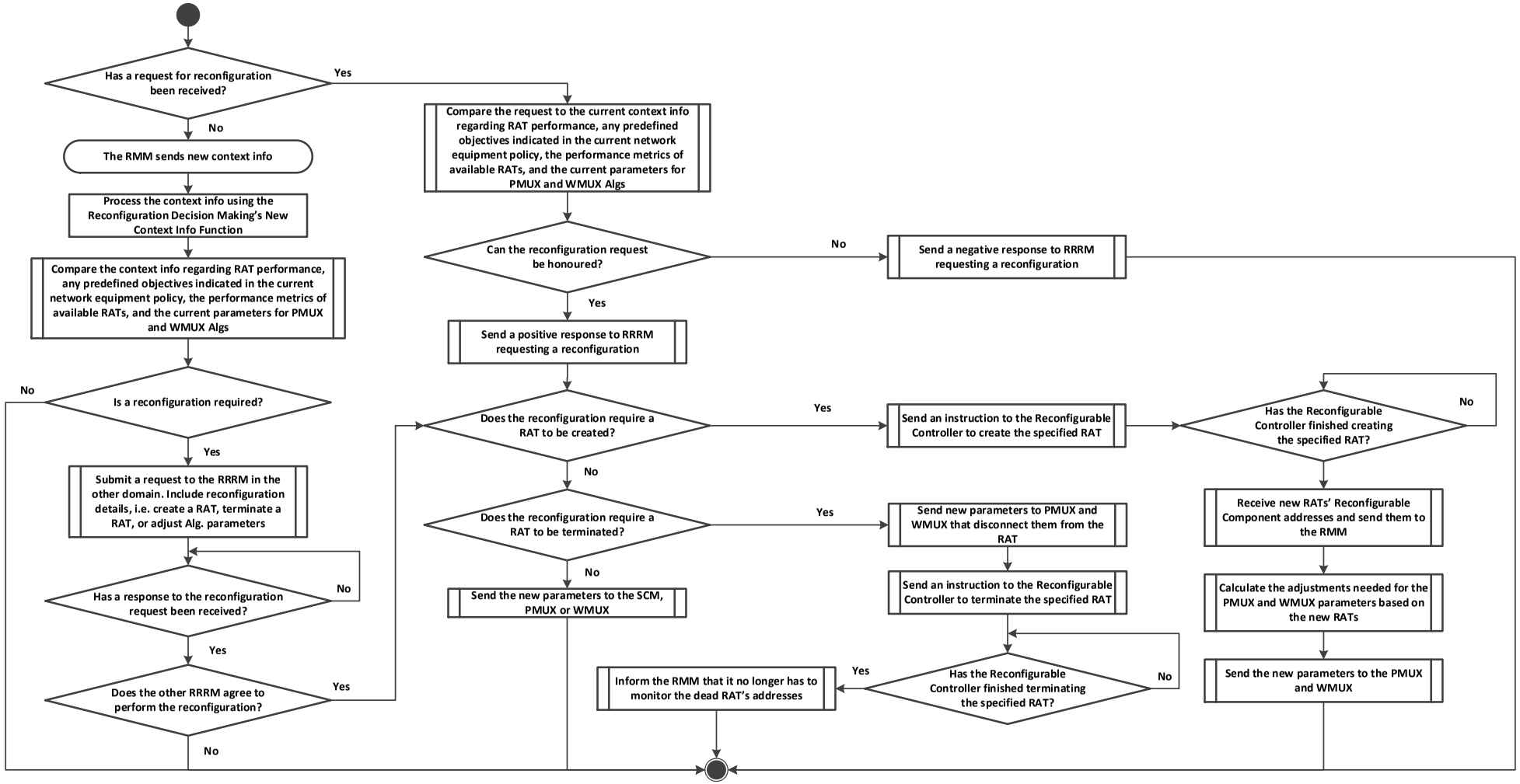


Figure 3.15: The internal operations of an RRRM whilst performing reconfigurations

3.7.3 Creating and Terminating Radio Access Technology Instances

Whilst the RRRM is responsible for determining when a reconfiguration must take place, it is the RMM which provides the context information needed to make the decision and the RC which implements the decision.

The purpose of the RMM is to gather context information from Reconfigurable Components in RATs, and to compile that information into a report which is sent to the RRRM. As there are no restrictions regarding the functionality implemented in Reconfigurable Components, the types of context information supplied by each Component cannot be predefined and are left to the developer to determine during implementation. Examples of potential types of context information include: the time taken for a Reconfigurable Component to process a set of inputs, the number of erroneous packets detected, the current throughput, and the Received Signal Strength Indicator (RSSI) value.

After a RAT is created, the RRRM must provide the RMM with a set of addresses for each Reconfigurable Component in the new RAT. The purpose of these addresses is to assist the RMM in classifying context information received from Reconfigurable Components according to their RAT. Figure 3.16 provides a detailed description of the RMM's internal operations.

The context information provided by the RMM enables the RRRM to make the reconfiguration decision to either create or terminate a RAT. The RC is responsible for carrying out these operations. In order to create a RAT the RC must first obtain the relevant RAT policy from the RPM and instances of the relevant Reconfigurable Components from the RCAL. The RC must then assign each Reconfigurable Component a unique address and provide all the relevant connection information for the Component's routing table. Finally, the RC must inform the RRRM that the RAT was successfully created, and provide a list of the addresses of the new RAT's Reconfigurable Components. Communications between the logical components involved in the creation of a RAT, and the monitoring and gathering of context information thereafter, are defined in Figure 3.17.

Should the RRRM instruct the RC to terminate a RAT, the RC needs simply to instruct each Reconfigurable Component within that RAT to complete its current task and then to terminate itself. When terminating a RAT the RRRM's Reconfigurable Decision Making Algorithm must decide how to handle the packets and signals currently being processed by the RAT. There are two potential approaches

to this problem: the first involves the Algorithm instructing the RC to immediately terminate the RAT and to discard any packets and signals currently being processed by the RAT, the second requires the Algorithm to first disconnect the PMUX from the RAT and allow the RAT to finish processing all packets and signals before instructing the RC to terminate the RAT. Both approaches involve compromises: the first results in packet loss but benefits from a rapid reconfiguration process, whilst the second results in a substantial delay in the reconfiguration process, but no packet loss. The choice between these approaches depends on the services being provided by the network operator and their respective QoS parameters.

Figure 3.18 demonstrates the communications between logical components involved in terminating a RAT, and Figure 3.19 defines the internal operations of the RC.

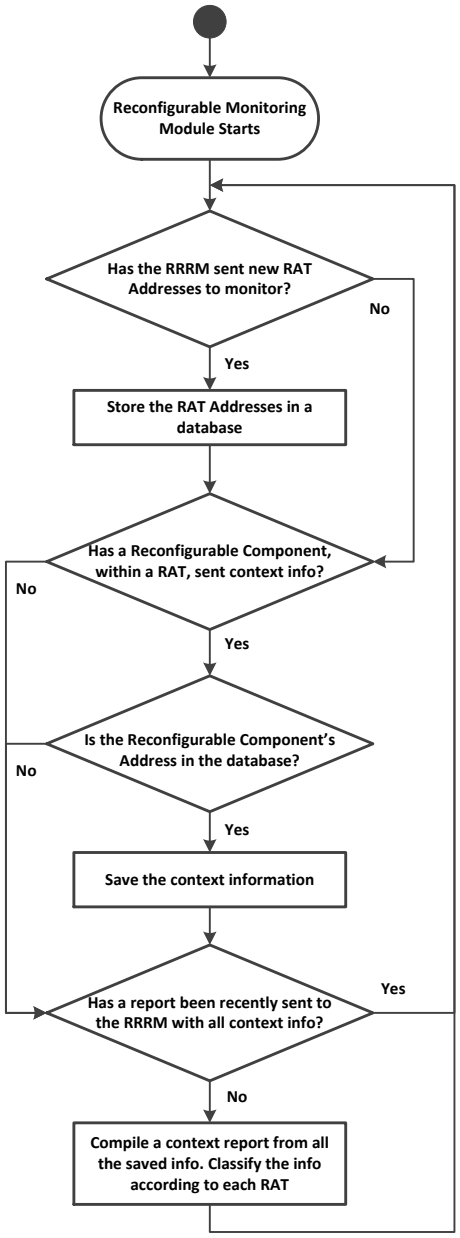


Figure 3.16: The internal operations of the RMM

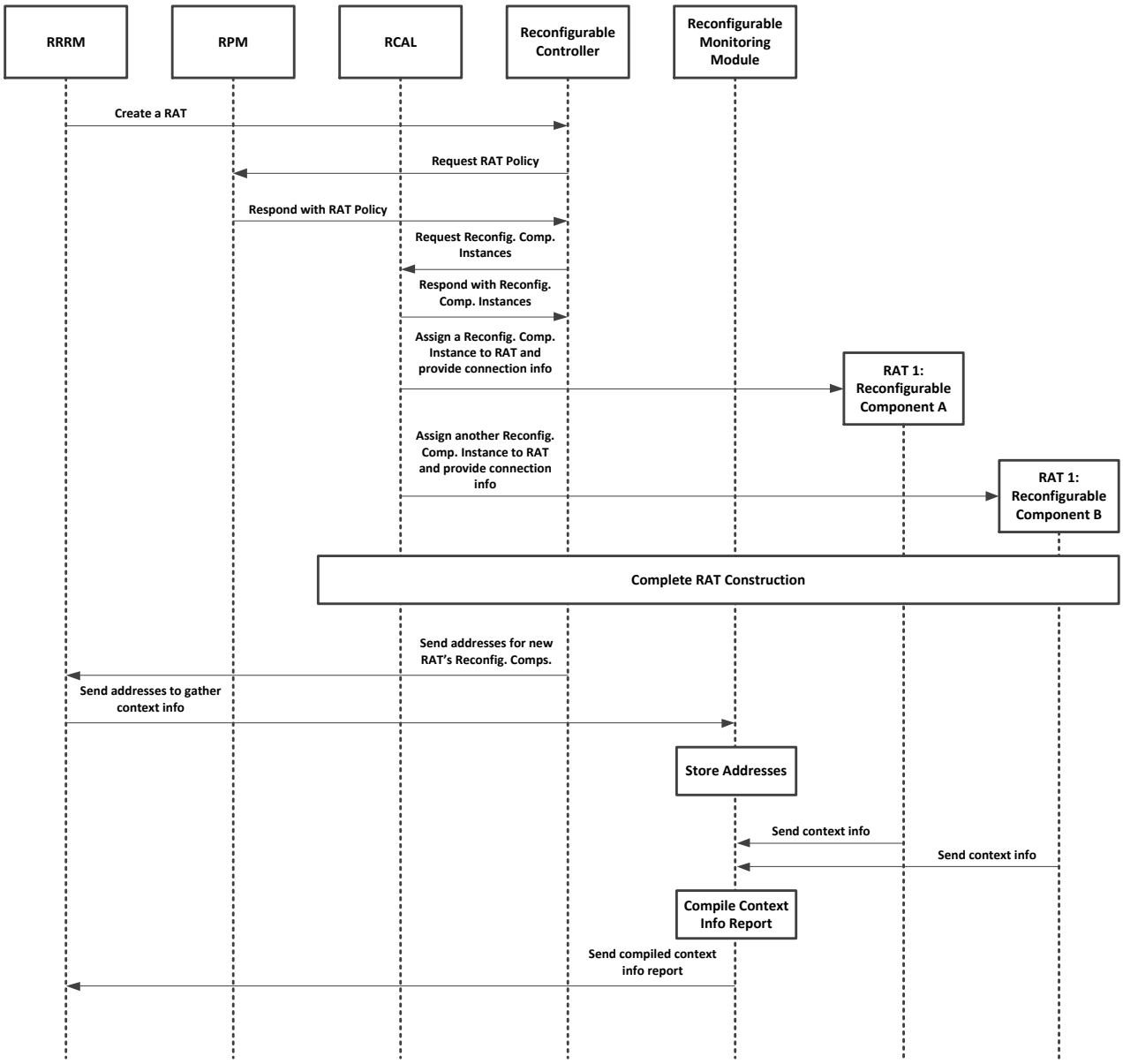


Figure 3.17: Communications between the RC, RMM and other logical components during RAT creation

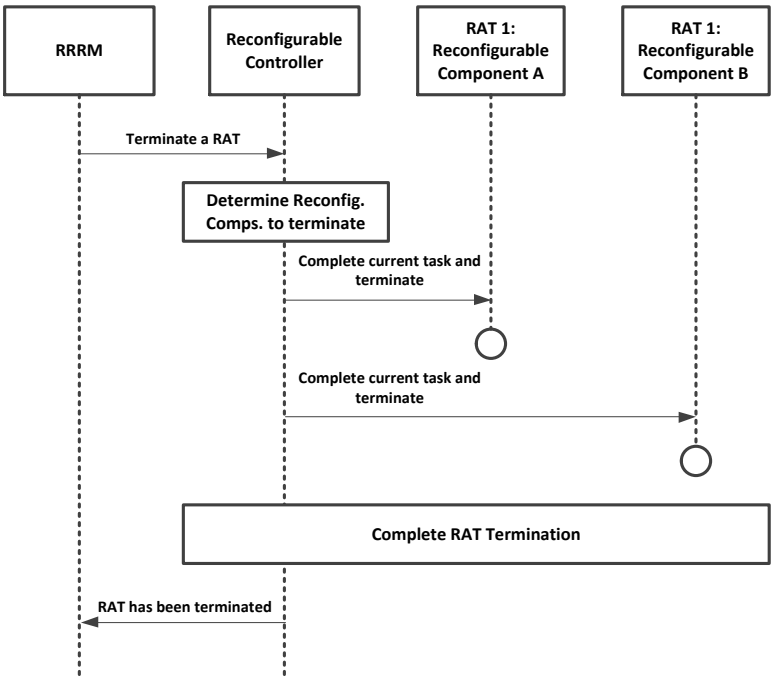


Figure 3.18: Communications between the RC, RMM and other logical components during RAT termination

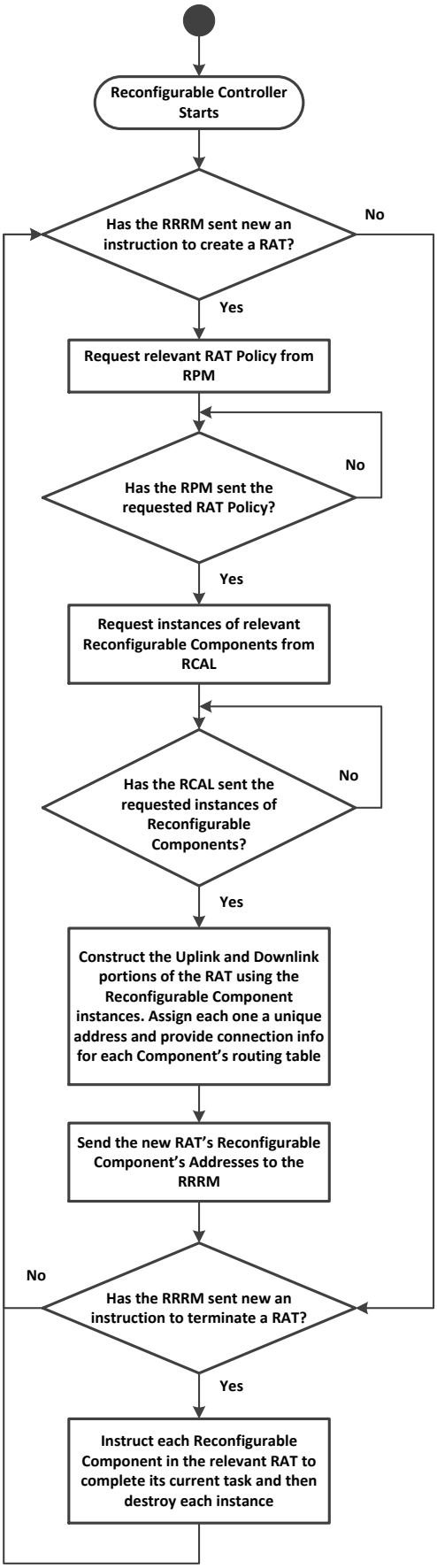


Figure 3.19: The internal operations of the RC

3.8 Inter-RAT Traffic Scheduling

The PMUX is responsible for scheduling Network Layer traffic between RATs based on their QoS requirements or some other predefined criteria. The method used to schedule traffic is implemented by the Reconfigurable Packet Multiplexing Algorithm as defined in Figure 3.20. The RRRM can control the flow of traffic between RATs by selecting a different Reconfigurable Packet Multiplexing Algorithm to implement or by adjusting the current implementation's parameters. This is usually performed after a new RAT has been created, and before an old RAT is terminated. Obviously this does require that the RRRM's Reconfigurable Decision Making Algorithm possesses predefined knowledge regarding the methods used to schedule traffic in each implemented Reconfigurable Packet Multiplexing Algorithm.

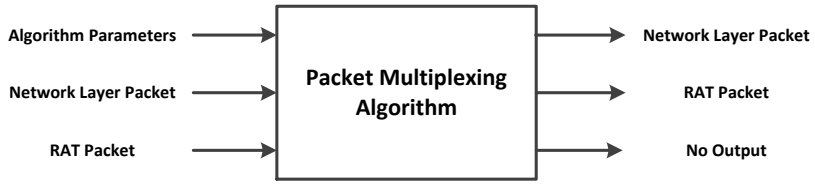


Figure 3.20: The Packet Multiplexing Algorithm

The PMUX, like the RRRM and SCM, acts as a wrapper to the Packet Multiplexing Algorithm, and it also provides the interfaces for exchanging messages with other logical components in the CRRSA. Although developers may make use of any packet routing technique to implement the Reconfigurable Packet Multiplexing Algorithm, the PMUX requires the Algorithm to implement a function to process packets received from the Network Layer, and a function to process RRS management messages received from logical components in the same domain.

As the length of time taken by RATs to process packets and signals depends entirely on the functions implemented in Reconfigurable Components, and as these values cannot be predetermined, the PMUX buffers all packets received from the Network Layer until they are requested by Reconfigurable Components in each RAT.

Figure 3.21 provides a detailed specification of the internal operations performed by the PMUX, as well as how the Reconfigurable Packet Multiplexing Algorithm is incorporated into the process.

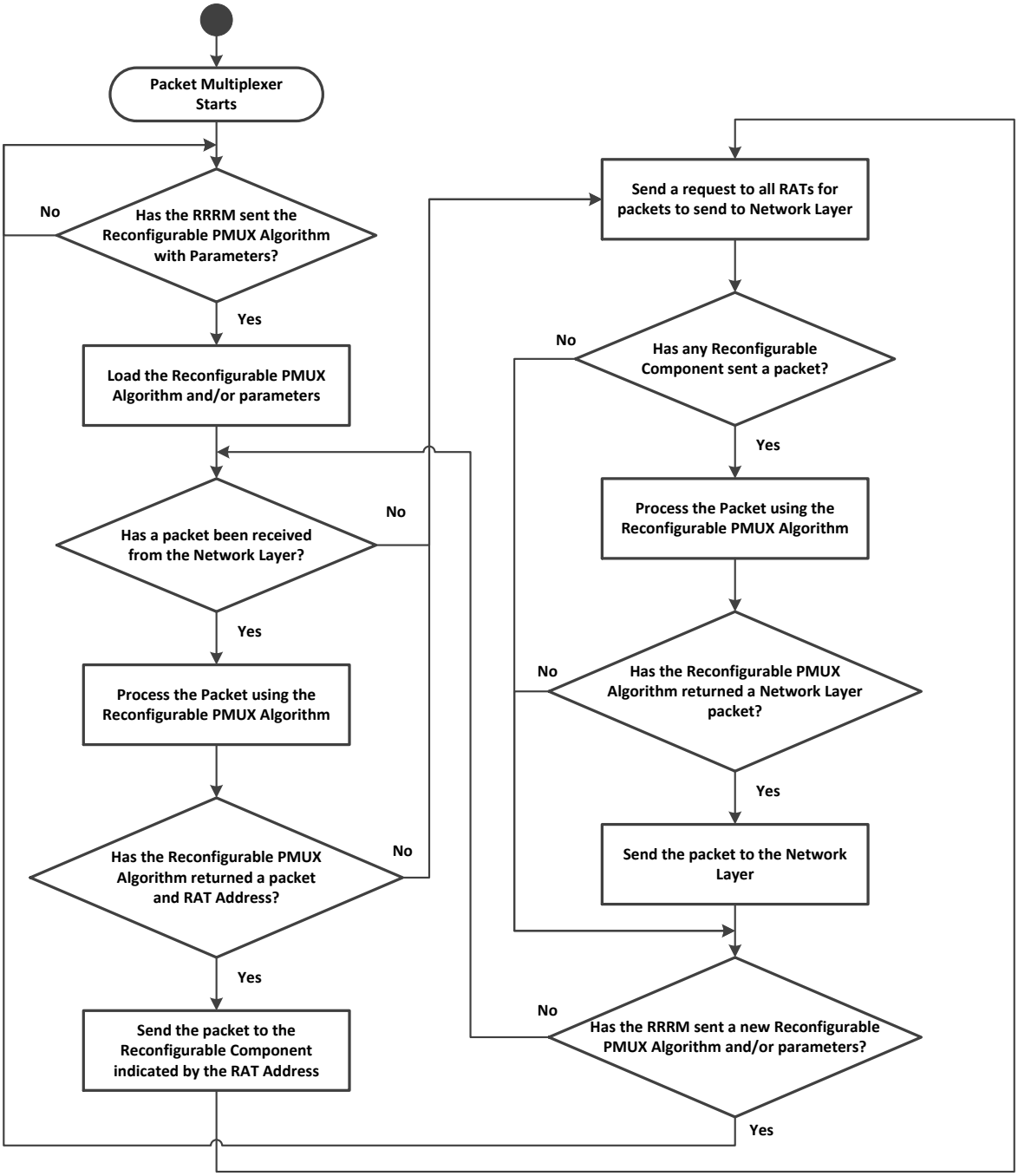


Figure 3.21: The internal operations of the PMUX

3.9 Interfacing RATs with SDR Hardware

Interfacing RATs defined in software with generic reusable SDR hardware platforms presents a number of significant challenges:

- The signals produced by each of the RATs need to be combined before they can be transmitted by the SDR platform. This process whereby this is accomplished must be carefully selected, as it has a direct effect on the performance of all RATs, and must also be flexible enough to incorporate different duplexing schemes and advanced signal processing techniques such as spatial multiplexing. Prior to the separation of the signals at the receiver, the use of timing and frequency synchronisation techniques may be required.
- As different RATs require access to different radio resources and transmission and reception parameters in the SDR platform, an arbitration mechanism must be provided to respond to requests from RATs and to attempt to optimise the distribution of radio resources as well as to firm up on which transmission and reception parameters are to be implemented.
- As each RAT and the SDR platform processes signals at different rates, a buffering mechanism must be implemented in order to harmonise the rates of both types of systems.

The WMUX is the logical component in the CRRSA responsible for providing these functions. The Reconfigurable Waveform Multiplexing Algorithm determines the method and signal processing techniques used by the WMUX during its operations. The WMUX acts as a wrapper to the Reconfigurable Waveform Multiplexing Algorithm, and provides a standardised set of interfaces for RATs to send and receive signals, for those signals to be sent to and received from the SDR API, to the obtain wireless channel quality measurements from the SDR API and relay them to requesting RATs, and to provide RATs with the ability to request changes to transmission and reception parameters and to relay those changes to the SDR API.

The Reconfigurable Waveform Multiplexing Algorithm, defined in Figure 3.22, is required to provide a function capable of multiplexing all signals received from RATs into a single waveform, or multiple in the event that multiple antennae are used, which can be transmitted by the SDR platform, as well as a function to demultiplex the waveform received by the SDR platform into multiple signals which are provided

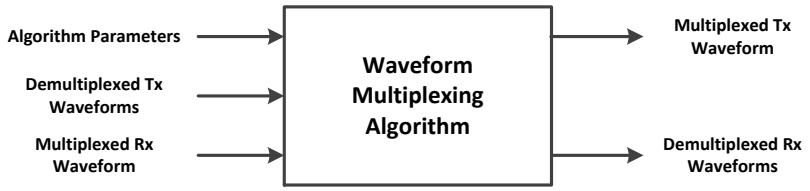


Figure 3.22: The Waveform Multiplexing Algorithm

to each RAT. As mentioned above, these functions may also need to perform timing and frequency synchronisation of signals received if the SDR platform is not able to adequately perform this function.

The WMUX provides the RRRM with the ability to adjust the manner in which its network element interacts with the wireless environment. Subject to the RRRM's Reconfigurable Decision Making Algorithm having been developed to be compatible with the methods implemented in the current Reconfigurable Waveform Multiplexing Algorithm, the RRRM can use this functionality to issue and adjust spectrum allocations for each existing RAT.

Figure 3.23 provides a detailed description of the WMUX and its internal operations.

The SDR API is the final logical component in the CRRSA. It is responsible for providing a layer of abstraction between any SDR hardware platform and the WMUX. The CRRSA only specifies the interface which must be provided by the SDR API, as its internal operations are implementation specific and are also highly dependent on the SDR hardware platform. In the event that a particular SDR hardware platform is incapable of supporting a specific operation, e.g. a particular duplexing scheme or spatial multiplexing, a compatible WMUX Reconfigurable Algorithm must be implemented. This approach allows virtually any SDR hardware platform to be reused and seamlessly integrated with the CRRSA.

The SDR API is required to provide the WMUX with the ability to transmit and receive signals on a single or multiple antennae, specify the duplexing scheme to be implemented, obtain wireless environment quality measurements, and adjust transmission and reception parameters. Communication between the WMUX and the SDR API, and between the WMUX and RATs, is defined in Figure 3.24.

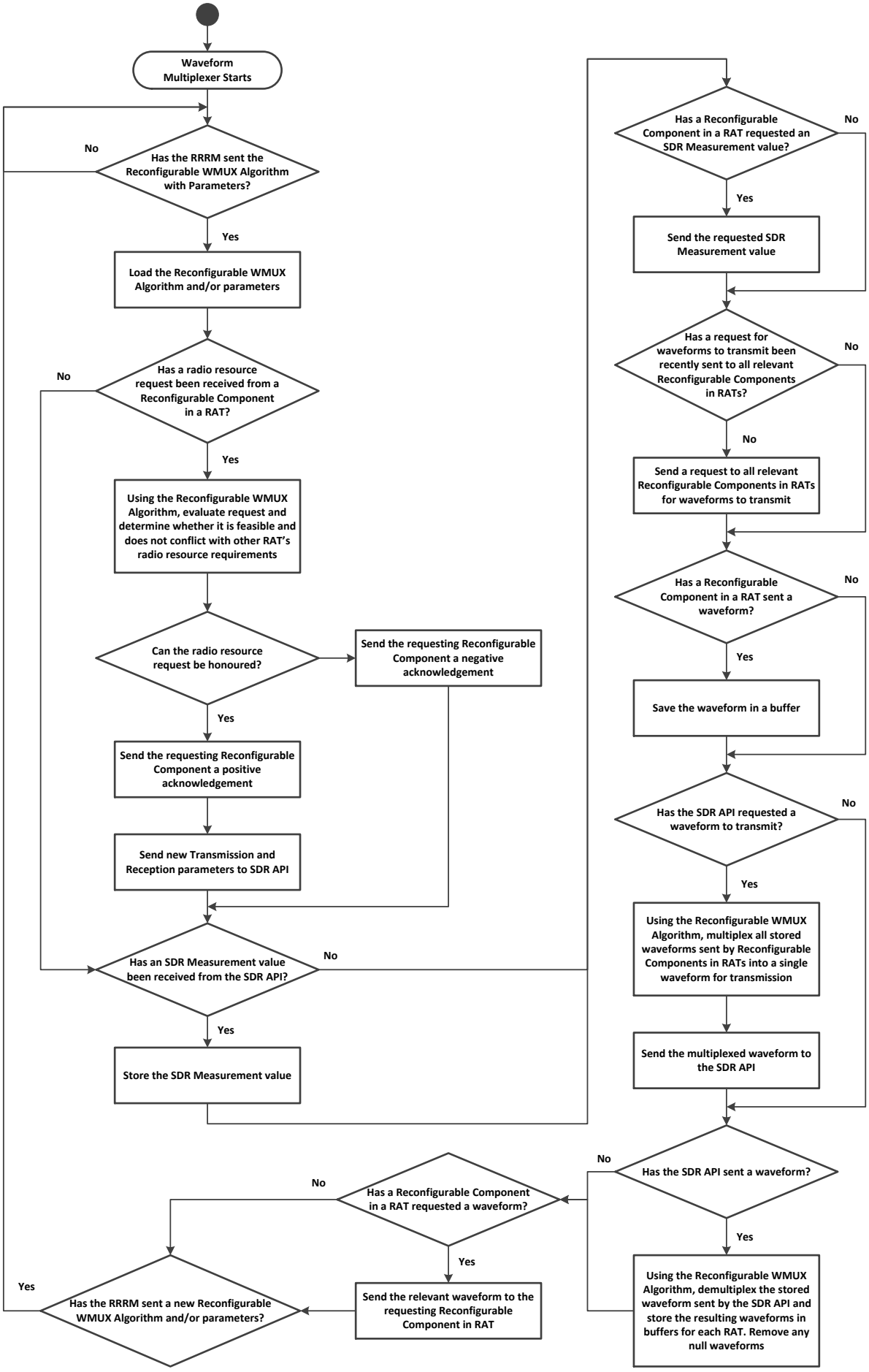


Figure 3.23: The internal operations of the WMUX

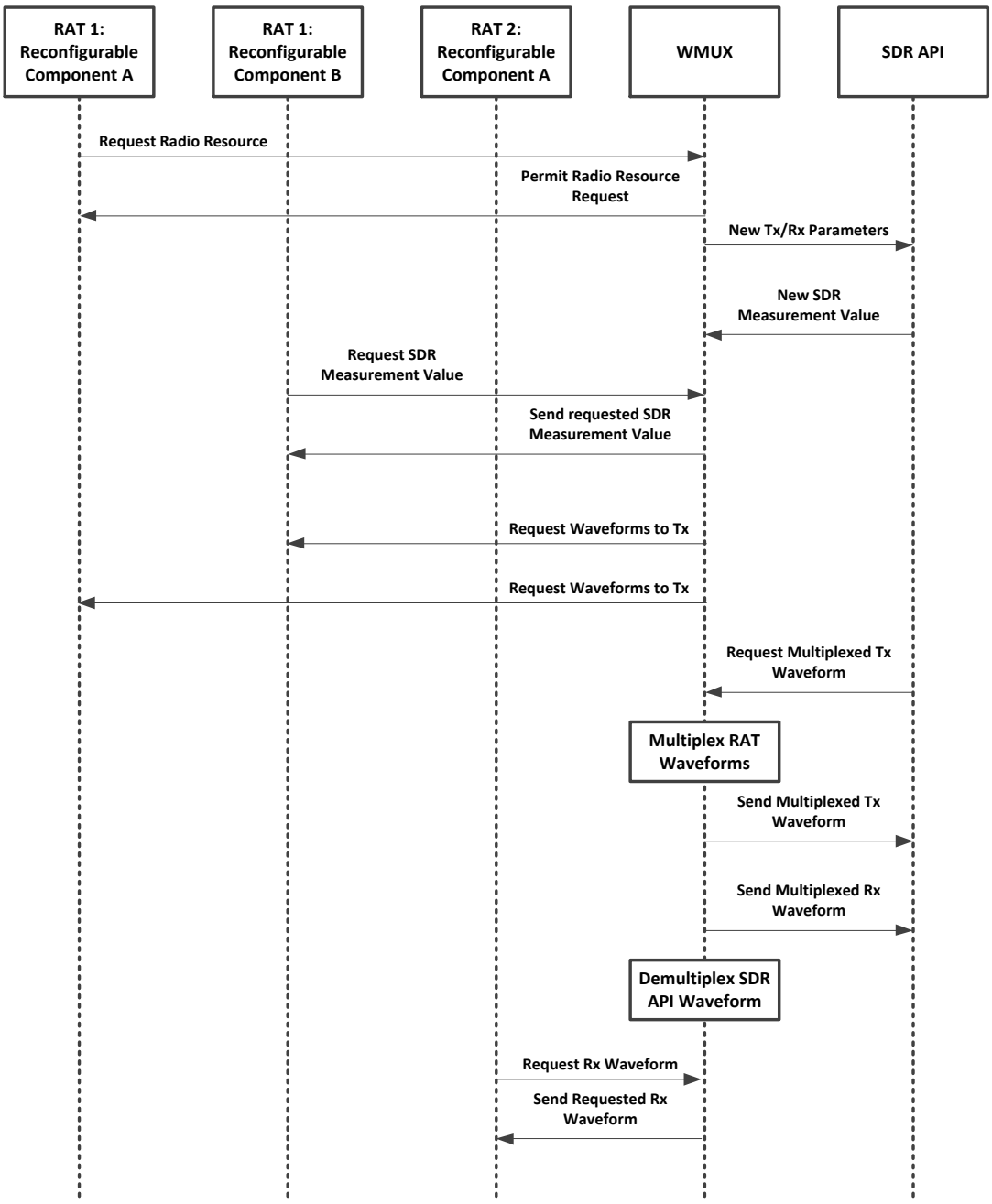


Figure 3.24: Communications between the SDR API, WMUX, and RATs during transmission and receiving operations

3.10 Summary

The CRRSA is a complete RRS solution which provides a reconfigurable management system, an operating environment, and a radio application development and implementation framework. Chapter 2 defines a set of comprehensive functional requirements for these components of a unified RRS solution, which are encapsulated by the CRRSA in a set of logical components interconnected via standardised interfaces. The following provides a summary of these functional requirements and explains how they are fulfilled by the CRRSA:

- **Operating Environment Functional Requirements**: relate to the supporting infrastructure of an RRS, which enables its reconfigurable management system and the implemented RATs to successfully complete their tasks. The CRRSA meets these requirements as follows:
  - *Secure Communications*: The CRRSA's reconfigurable management systems within the Core Network, Subscriber, and Access Network domains are able to securely communicate with one another using their SCMs. These SCMs can be upgraded or modified in order to improve their functionality as new security threats are identified.
  - *Interface with Network and higher layers of the Protocol Stack*: The SCMs provide all reconfigurable management systems with an interface to the Network Layer within their network elements, which in turn interfaces with each implemented RAT via the PMUX.
  - *Interface with SDR hardware platforms*: The SDR hardware platform interfaces with the implemented RATs using the CRRSA's WMUX and the SDR API in each network element.
- **Reconfiguration Management Functional Requirements**: specify the functionality needed to manage and control reconfiguration operations within a RRS. The CRRSA realises these requirements as follows:
  - *Cognitive Reconfiguration Capabilities*: The CRRSA's RRRM is responsible for making reconfiguration decisions based on context information, and then instructing the RC to execute those decisions. The RRRM's decision making algorithm can be upgraded or modified to improve the process of making reconfiguration decisions and the quality of those decisions.

– *Monitor and Collect Context Information*: Each Reconfigurable Component in an implemented RAT supplies context information to the RMM, which in turn compiles the information into context information reports that are periodically sent to the RRRM. The type of context information to be supplied is not defined by the CRRSA, thereby enabling developers to choose which information should be sent to the RMM based on the functionality each Reconfigurable Component provides.

– *Policy-based Management*: The CRRSA provides two types of policies, namely Network Equipment policies and RAT policies. Network Equipment policies specify which RATs may be implemented, and the names of, and default parameters for, the Reconfigurable Algorithms to be used in each network element. RAT policies specify the performance metrics, as well as Reconfigurable Components used to implement RATs. The RPM in each network element manages all policies, and issues them to the RRRM and RC upon request.

– *Validation of Software and Policies*: The CRRSA's RPM and RCAL validate all policies, Reconfigurable Components, and Reconfigurable Algorithms prior to their use in reconfiguration operations.

– *Dynamic Runtime Reconfiguration of RRSs*: The CRRSA is specifically designed to support dynamic runtime reconfiguration operations, and the RC can create and terminate RATs at runtime based on reconfiguration instructions received from the RRRM.

– *Joint Radio Resource Control and Inter-RAT Traffic Scheduling*: The RRRM in each network element controls radio resources and inter-RAT traffic scheduling by creating RATs, terminating RATs, and/or modifying the parameters of Reconfigurable Algorithms in the PMUX and WMUX. Joint radio resource management is achieved through Reconfigurable Algorithm parameters defined in each network element's Network Equipment policy, which in turn is defined by the Network Operator and is tailored to each network element's Reconfigurable Decision Making Algorithm.

– *Dynamic Spectrum Access*: The WMUX provides the RRRM with the ability to adjust the transmission parameters of the SDR hardware platform via the SDR API. This functionality can be used to adjust spectrum allocations for each implemented RAT.

– *OTA Download of Software Upgrades and Policies*: The RPMs and RCALs in the Subscriber and Access Network domains periodically request new

policies as well as Reconfigurable Components and Algorithms from the RPM and RCAL in the Core Network domain. These new policies, and the Reconfigurable Components and Algorithms, are transmitted between the Subscriber and Access Network domains using existing RATs.

  - *CPC Solution*: Each Network Equipment policy specifies at least one default RAT, which the RRRM must create after its network element is booted. The RRRM can also determine which existing RAT carries reconfiguration management messages by modifying the parameters belonging to the Reconfigurable Packet Multiplexing Algorithm.

- **Radio Applications Functional Requirements**: define a RRS's radio application development and implementation framework which is capable of dynamically implementing any legacy, current, or future RAT in a scalable and flexible manner. The CRRSA fulfils these requirements as follows:

  - *Radio Application Development Framework*: The CRRSA defines a logical architecture for Reconfigurable Components, and standardised interfaces to interconnect Reconfigurable Components when constructing specific RATs. Radio application developers can use this framework to develop new RATs which can in turn be used in any CRRSA implementation.

  - *Support any RAT using RPSs*: Each RAT is constructed from numerous Reconfigurable Components, based on connection instructions contained in RAT policies. This approach enables any legacy, current, or future RAT to be implemented in a flexible and scalable manner. In addition, this approach also enables each Reconfigurable Component to potentially be reused in implementing multiple separate RATs, thereby reducing developing costs and improving software reliability.

To prove the viability of using the CRSSA to model and implement RATs, the following chapter examines two modern RAT standards and derives two corresponding sets of Reconfigurable Components. The CRRSA and the Reconfigurable Components mentioned above are implemented in Chapter 5; initially in a simulated wireless environment and then in a real wireless environment using a commonly-adopted SDR hardware platform, in order to prove that the CRSSA is both capable of performing reconfiguration management operations and of implementing modern RATs in a practical and realistic wireless environment.

Chapter 4

Modelling Modern Radio Access Technologies

The previous chapter presented the CRRSA model and described how the functional requirements defined in Chapter 2 were incorporated into a reconfigurable management system and operating environment capable of dynamically implementing RATs at runtime. To prove that the CRRSA model is both viable and capable of executing its operations in a functional environment, Chapter 5 presents an implementation of the CRRSA using a combination of custom simulation software and an SDR hardware platform.

The CRRSA implementation demonstrates the functional capabilities of the CRRSA, and also proves that the CRRSA is capable of implementing one or more RATs which in turn are able to transmit and receive a communications signal containing data over a wireless link without errors. The purpose of this chapter is to define a set of Reconfigurable Components, and to model two separate RATs using those Components. This forms part of the CRRSA implementation in the subsequent chapter and proves the viability of the Architecture.

All RATs must contain specific functions which are required to establish and maintain a wireless connection. Although different RATs may implement these functions using different digital signal processing techniques and link control algorithms, their core functionality remains the same. Many communication simulation packages exploit this fact by providing a standardised set of building blocks which can be combined to create virtually any RAT [22,94,95]. The concept of RPSs is also based upon this approach. As the successful implementation of RATs in the CRRSA is determined by whether it is capable of transmitting and receiving an error-free signal using such RATs, the functionality included in the Reconfigurable Components used

to construct these RATs was limited to those elements required to achieve this goal, i.e. the essential functions within each communication standard's MAC and Physical Layers.

The two RATs modelled in this chapter are derived from two modern RAT standards; the IEEE's 802.16 standard [96], or WiMAX as it is more commonly known, and the 3GPP's Long Term Evolution (LTE) standard [97]. These standards were selected because they employ complex modern signal processing techniques and algorithms to perform their functions and they are the most advanced established RATs in existence. Many researchers have already extensively modelled and implemented the 802.16 and LTE standards using communication simulation packages, but such implementations primarily focus on testing and measuring the performance of each standard's air-interface and not on the dynamic reconfiguration of RATs at runtime.

By developing two RAT models which contain the generic functions used by all RATs to establish and maintain a wireless connection, and by utilising the same signal processing techniques and algorithms as those employed in 802.16 and LTE, it can be demonstrated that the CRRSA is both future-proof and capable of implementing any RAT via dynamic reconfiguration operations which are performed at runtime.

Throughout this chapter it is assumed that the reader is well versed in the fields of signal processing and communications systems, and therefore these topics, and the implementation thereof, are not focused on in significant detail. Additional information on such topics can be found in [14, 16, 95, 98].

The rest of this chapter proceeds as follows: Sections 4.1 and 4.2 provide a brief overview of the 802.16 and LTE standards, and then examine those functions, algorithms, and signal processing techniques employed by each standard which are considered necessary to successfully transmit and receive a communications signal carrying data without any errors. Section 4.3 demonstrates how each standard's fundamental components are decomposed and then modelled in Reconfigurable Components. Finally, Section 4.4 describes how the two different RATs, derived from 802.16 and LTE, are created using a combination of the modelled Reconfigurable Components.

4.1 The IEEE 802.16 Radio Access Technology

Unlike traditional cellular mobile telephony RAT standards such as GSM or UMTS, the IEEE 802.16 RAT standard was initially developed as a point-to-multipoint line of sight (LOS) wireless networking system with data transmission being its primary function. Development of the standard began in 1998 and the first version was published in 2001. Subsequent revisions to the standard incorporated non line of sight (NLOS) capabilities in 2003 and support for mobility in 2005. In 2007 the mobile version of the standard (802.16e) was accepted by the ITU for inclusion within IMT-2000 [36], thereby classifying it as a 3G communication standard. (See [15, 22, 24, 99])

In 2006 the IEEE's 802.16 working group determined that a revision of the 802.16e standard was necessary in order to meet the requirements put forth by the ITU for IMT-Advanced [100]. The 802.16m version of the standard was published in May 2011 [101], and was one of only two standards accepted by the ITU for inclusion within IMT-Advanced in January 2012 [36].

Although the 802.16 standard is commonly referred to as WiMAX, this is misleading. The IEEE 802.16 standard only specifies the air-interface, which includes the Medium Access Control (MAC) and Physical Layers. The WiMAX Forum, founded in 2001, publishes a separate standard [102] which specifies the network architecture used in conjunction with the 802.16 air interface, and it is the combination of these two standards which constitute the WiMAX communication system [12, 22, 24].

The 802.16 standard provides telecommunications equipment vendors and network operators with the ability to customise 802.16 implementations based on their application and deployment requirements. The standard specifies multiple options for the MAC layer's architecture, five different physical layers, and enables transmission over a wide range of spectrum [12, 15, 22, 24, 99]. Due to these numerous options, the WiMAX Forum offers interoperability and performance certification for two WiMAX equipment configurations or profiles, namely a fixed configuration which utilises OFDM, and a mobile configuration based on OFDMA [12, 22, 24, 103].

The following subsections provide an overview of the 802.16 standard's MAC and Physical layers. As previously mentioned, the focus of these discussions is on those functions in each of the layers which are considered critical to establishing and maintaining a functional wireless link. It should also be noted that the parameters

specified and used in modelling the Reconfigurable Components are based on the 2009 version of the 802.16 standard [96]. Although different versions of the standard do contain slightly different link control algorithms and physical layer parameters, the underlying techniques and technologies have largely remained the same since the publication of the 802.16e version in 2005. Additional information relating to 802.16 can be found in [12, 15, 22, 24, 99].

4.1.1 The 802.16 MAC Layer

The 802.16 MAC layer is connection oriented and designed to support a point-to-multipoint (PMP) communication topology. As depicted in Figure 4.1, the MAC layer is divided into three sublayers, namely the Service-Specific Convergence sublayer (CS), the Common Part sublayer (CPS), and the Security sublayer. The Security sublayer provides encryption, authorisation, and encryption key exchange procedures.

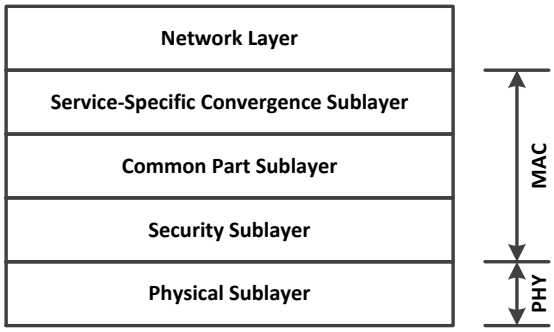


Figure 4.1: The IEEE 802.16 Protocol Stack [12, 15, 24, 96, 99]

The CS provides a layer of abstraction between the core MAC functions implemented by the CPS and the different protocols that could be used in the Network layer. Each packet received from the Network Layer, referred to as a MAC Service Data Unit (SDU), is classified according to its protocol and QoS requirements, and is assigned a Connection Identifier (CID). As the MAC and Physical layers of the 802.16 standard do not provide any form of Network layer addressing, the CID acts as a MAC or Physical layer address which can be mapped back to a Network layer address. The CID is a unique 16-bit unidirectional (i.e. either Uplink or Downlink) identification number which is assigned to a logical connection between

a Base Station and Mobile Device. In the event that there are different traffic flows with separate QoS requirements, a new CID is assigned to each one.

To improve traffic throughput and latency, the CS also provides Packet Header Compression (PHC), which removes repetitive sections of Network Layer packet headers at the transmitter and replaces them at the receiver. The 802.16 standard provides classification and PHC specifications for two primary types of Network Layer packets: Internet Protocol (IP) and Asynchronous Transfer Mode (ATM).

The CPS contains all of the major MAC functions which are required to establish and maintain a mobile wireless connection between a Base Station and a Mobile Device. A connection is established using the Ranging function, which allows a Mobile Device to determine the required timing, frequency, power, and other channel parameters. Ranging is also used to determine when a Mobile Device handover between Base Stations should take place.

Once a connection has been established, three logical signalling connections are established and assigned specific CIDs, these are: the Basic Connection which carries short time-critical MAC signalling messages, the Primary Management Connection which carries longer delay-insensitive MAC signalling messages, and the Secondary Management Connection which carries network and higher layer signalling messages.

Quality of Service (QoS) is a major challenge in wireless communications due to the constant varying nature of the environment [99]. Although the 802.16 standard does not define the scheduling algorithm to be used in implementations [15, 24, 99], it does specify a set of scheduling services which define how the CPS should manage different types of traffic with specific QoS requirements. Each connection's minimum and recommended QoS requirements are defined by their Service Flow Identifier (SFID). The SFID, together with the CID, enables the scheduler to determine which connections should be prioritised over others.

The scheduler is only implemented in the Base Station, and it determines how much bandwidth is allocated to each Mobile Device in the uplink and downlink. A Mobile Device may request additional bandwidth; however this is only be allocated if the scheduler can spare the resources without compromising the performance of other higher priority connections.

Adaptive modulation and coding is also controlled by the CPS, and there are pre-defined combinations of modulation and channel coding parameters and techniques which may be used during transmission. These combinations, which are referred to

as burst profiles, can be varied via negotiation between the Mobile Device and the Base Station in order to improve the performance of each connection.

After the CS has assigned a CID to a MAC SDU and performed PHS, the CPS must encapsulate that SDU in a MAC Protocol Data Unit (PDU) which is passed to the Physical Layer. Depending on the size of the SDU, there are two different methods which may be used to construct a PDU. If the SDU is smaller than the PDU, then multiple SDUs can be concatenated and carried in a single PDU, otherwise if the PDU is smaller than the SDU, the SDU must be segmented into fragments, each of which are carried via their own PDU.

Figure 4.2 presents the Generic MAC Header for PDUs, which contains the length of the payload, the CID, an indication as to whether a Cyclic Redundancy Check (CRC) is appended, the type of the PDU, the CRC value for the header, and any encryption information. In the event that fragmentation or concatenation are utilised during the construction of a PDU, a set of special sub-headers are used to demarcate the position of each SDU. Aside from the Generic MAC PDU used to convey all data and most of the signalling messages between a Mobile Device and Base Station, the 802.16 standard defines a special MAC PDU to be used when making bandwidth requests.

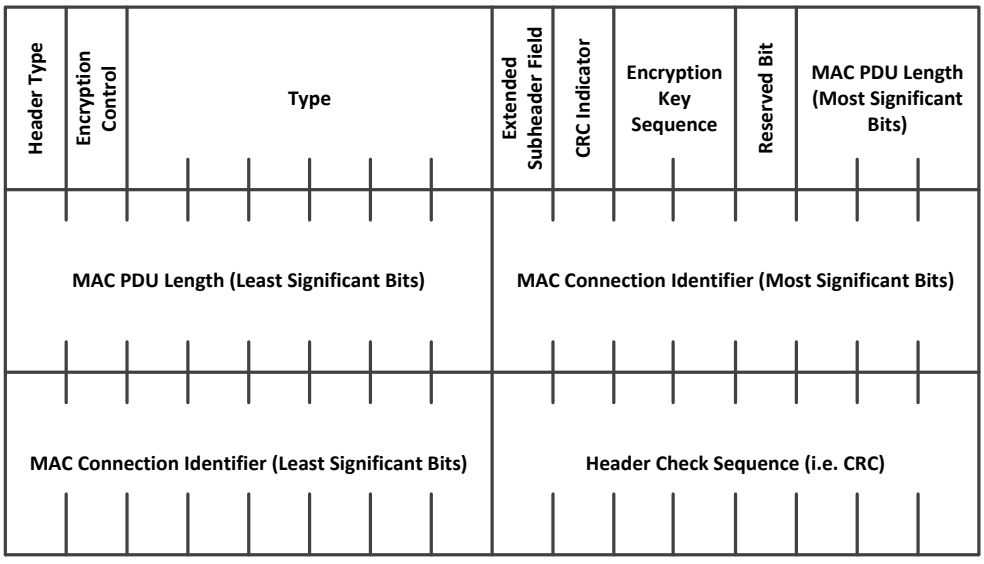


Figure 4.2: The IEEE 802.16 Generic MAC Header Format [12,15,22,24,96,99]

In addition to the functionality described above, the CPS also performs power saving operations for Mobile Devices, provides a Hybrid Automatic Repeat Request

(HARQ) algorithm, and assists the Network Layer in making and implementing handover decisions and other mobility management operations.

4.1.2 The 802.16 Physical Layer

The 802.16 standard contains five different Physical Layer specifications, viz. WirelessMAN-SCa (single carrier), WirelessMAN-OFDM, WirelessHUMAN, Wireless-OFDMA, and WirelessMAN-Advanced [12, 15, 22, 24, 99]. This subsection focuses on the functionality provided by the Wireless-OFDMA specification which was first introduced in 2005 as part of the 802.16e standard, and was later revised in 2009. Although the WirelessMAN-Advanced specification is more recent, the two physical layers utilise the same basic signal processing techniques and transmission technologies with different parameters and implementation requirements.

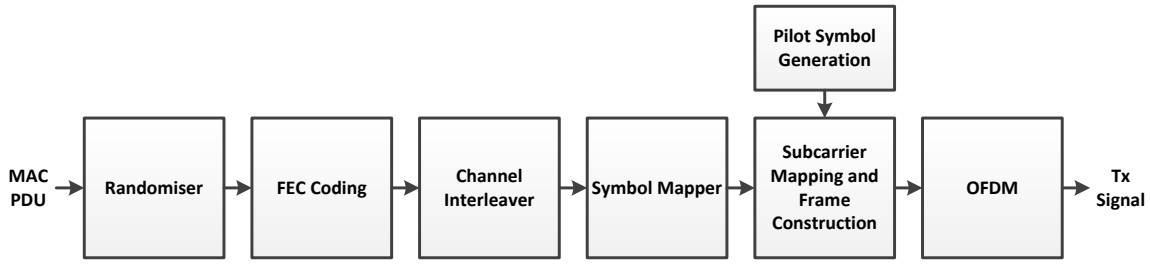


Figure 4.3: The IEEE 802.16 Physical Layer Primary Functions [12, 22, 24, 96]

Figure 4.3 depicts the primary functions of the Wireless-OFDMA physical layer. As is the case with most RAT standards, the 802.16 standard only specifies the functions for the transmission portion of the physical layer, leaving the receiving functions up to the implementer [22]. The first stage of the physical layer involves performing a randomisation operation on each MAC PDU. The Randomiser provides basic physical layer encryption for the transmitted signal, and is implemented using a shift-register with a predefined initialisation sequence.

Forward Error Correction (FEC) coding adds redundant information to each PDU, thereby providing a mechanism for correcting any errors introduced by short bursts of interference. The 802.16 standard specifies four different FEC coding schemes which may be used, including Convolutional Coding with tail-biting, Block Turbo Coding, Convolutional Turbo Codes, and Low Density Parity Check Codes. The

coding rate, the coding scheme used, and the parameters for each scheme are all defined based on the burst profile dictated by the MAC CPS.

The Channel Interleaver is defined by a set of equations unique to the 802.16 standard, and consists of two stages. The first stage adds frequency diversity and ensures that adjacent bits in the encoded PDUs are not mapped onto adjacent subcarriers during Frame Construction. The second stage of the interleaver ensures that adjacent bits in the encoded signal are not all mapped onto the same sections in the digital modulation constellation, thereby improving the robustness of the modulation scheme. (See [12, 22, 24])

The 802.16 standard provides three different symbol mapping, or digital modulation, schemes. These three digital modulation schemes are: QPSK, 16QAM, and 64QAM. Which scheme is used on each of the interleaved PDUs depends on the burst profile dictated by the MAC CPS. Scaling factors are specified for each scheme, thereby ensuring that the peak signal power for each scheme remains the same.

Pilot symbols are randomly produced and incorporated into the Physical layer frame to enable the receiver to perform channel estimation and equalisation. To improve the accuracy of the channel estimation, each pilot symbol is modulated according to its position in the physical frame, and each pilot symbol is transmitted with a power factor significantly higher than that of the average data symbol.

To create the Physical Layer Frame, the modulated data and pilot symbols must be mapped onto specific subcarriers in the frequency domain. The 802.16 standard provides a set of four Subcarrier Permutation Patterns which specify the optimum positions of each symbol based on the number of subcarriers and the channel conditions. The purpose of these permutations is to either achieve greater frequency diversity, by mapping each set of symbols to different subcarriers, or to map the symbols to the same subcarriers for beamforming and to exploit multiuser diversity [12, 22, 24].

Figure 4.4 presents the 802.16 WirelessMAN-OFDMA frame format using Time Division Duplexing (TDD); there is a separate frame format for Frequency Division Duplexing (FDD).

Physical Layer frames are composed of multiple sets of frequency subcarriers concatenated together in the time domain. Each frame begins with a Preamble which is used to perform time and frequency synchronisation at the receiver. The Frame Control Header is used to carry signalling information and also to specify the positions of

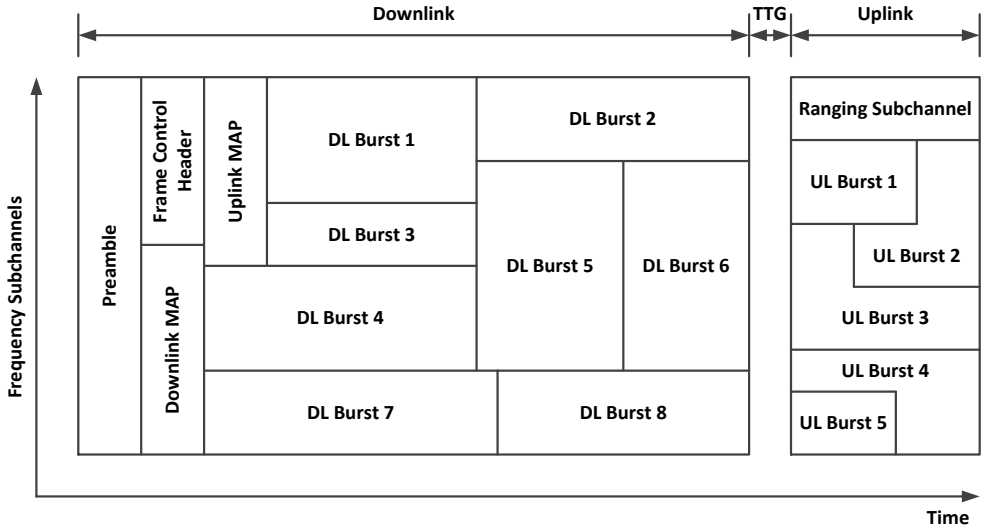


Figure 4.4: The IEEE 802.16 OFDMA TDD Frame Format [12, 22, 24, 96]

the Downlink MAP (DL-MAP) and Uplink MAP (UL-MAP). The DL-MAP and UL-MAP fields provide information regarding which subcarriers and time slots in the frame are allocated to each connection, as well as the burst profile for each one. The downlink and uplink frames are separated by a guard interval, and the uplink frame contains a field dedicated to Ranging. Finally, as mentioned previously, the Base Station defines how much bandwidth each Mobile Device may utilise, and this information is relayed to the Mobile Device using the UL-MAP field. Any changes to bandwidth allocations or burst profiles is reflected in changes in the UL-MAP and DL-MAP fields.

Orthogonal Frequency Division Multiplexing (OFDM) is a well-known modulation scheme which maps symbols onto orthogonally spaced frequency subcarriers using the Inverse Fast Fourier Transform (IFFT), thereby increasing the communication system's resilience against Inter-Carrier Interference (ICI) and Inter-Symbol Interference (ISI). To completely eliminate ISI, OFDM systems always include a guard interval between OFDM symbols. This guard interval is a cyclic extension of the OFDM symbol and is acquired by copying a section from the end of the symbol and then prepending it to the front of the symbol. In the 802.16 standard, OFDM plays a special role in mapping the data symbols relating to different connections onto subcarriers, thereby providing a multiple access scheme which enables multiple Mobile Devices to connect to a Base Station simultaneously.

After the symbols have been mapped to specific subcarriers using the Subcarrier Permutation Pattern, and they have been aligned in the time domain according to the frame format, OFDM is used to convert all mapped symbols into the frequency domain, thereby completing the frame construction process.

The 802.16 standard enables network operators to adjust the amount of bandwidth they utilise per Base Station by adjusting the number of subcarriers used when performing OFDM. This approach, known as Scalable OFDM (SOFDM), means that network operators can increase or decrease bandwidth available to each Base Station based on its anticipated load and the spectrum allocated to neighbouring Base Stations [12, 22, 24].

4.2 The 3GPP Long Term Evolution Radio Access Technology

The Long Term Evolution (LTE) RAT was published as part of the 3GPP's Release 8 group of specifications in 2008. The purpose of LTE is to ensure the continued competitiveness of the 3GPP's RAT standards by providing a new RAT capable of delivering reduced latencies, higher throughput, and increased traffic carrying capacity, at a lower cost when compared with existing systems. Although LTE does achieve these objectives, its throughput, spectral efficiency and other performance metrics do not meet the requirements for inclusion within the ITU's IMT-Advanced framework. (See [17, 19, 32])

Shortly before the publication of Release 8, the 3GPP began investigations into possible amendments which could be made to LTE in order for it to meet the requirements for inclusion in IMT-Advanced. The result of these investigations, and the subsequent standardisation process, is the LTE-Advanced RAT specification which was published as part of Release 10 in 2011 [32, 104]. In January 2012, after reviewing submissions made by 3GPP, the ITU formally accepted and incorporated the LTE-Advanced RAT standard into the IMT-Advanced framework [36].

The LTE standard is divided into a packet switched core network, called the Evolved Packet Core (EPC), and the air-interface, called the Evolved Universal Terrestrial Access Network (E-UTRAN). Unlike previous 3GPP RAT systems, all traffic in LTE is packet-based, and there is no logical division between telephony and data services, nor is there any provision for legacy circuit-switched systems. (See [17, 31, 38])

Compared to the 802.16 RAT standard reviewed in the last section, the LTE standard is substantially more complex. The reason for this is that the LTE standard was developed as a complete solution, which includes specifications for a core network and an air-interface, while 802.16 only specifies the air-interface. In addition, many of the procedures and techniques used throughout the LTE air-interface were driven by previous RAT standards, such as the use of logical and transport channels.

Despite the differences in complexity there are some material similarities between the two standards, particularly with regard to the functions and technologies incorporated in each standard. This is because all RATs must contain the same core functions needed to establish and maintain a mobile wireless connection.

The following subsections provide an overview of the E-UTRAN's protocol stack and each of its layers. As was the case with the 802.16 standard, the focus of these subsections is on those functions required to successfully establish and maintain a functional wireless link. In addition, the functions, techniques, and parameters specified and used to model the Reconfigurable Components are based on 3GPP's Release 9 [97] group of specifications, which refined the initial Release 8 version of LTE. Although the more recent Release 10 does contain the specifications for LTE-Advanced, the underlying technologies and techniques have remained largely the same, and hence no amendments were made to accommodate Release 10. Additional information relating to LTE can be found in [17, 19, 30–32, 38].

4.2.1 The LTE PDCP, RLC and MAC Layers

The E-UTRAN is divided into two planes containing data and control traffic, and four layers which contain the different functions required to successfully transmit and receive network layer packets over a wireless link. Figure 4.5 presents the data plane of the E-UTRAN protocol stack, which contains the Packet Data Convergence Protocol (PDCP) layer [105], the Radio Link Control (RLC) layer [106], the MAC layer [107], and the Physical layer [108]. The control plane includes an additional layer above the PDCP called the Radio Resource Control layer [109], which provides an interface between the core network and the air-interface for all functions related to mobility management and radio resource management.

The PDCP layer is very similar to the 802.16 standard's CS, and provides header compression for all Network layer packets based on the Robust Header Compression Algorithm used in many other RATs [17], for example in W-CDMA. The PDCP is

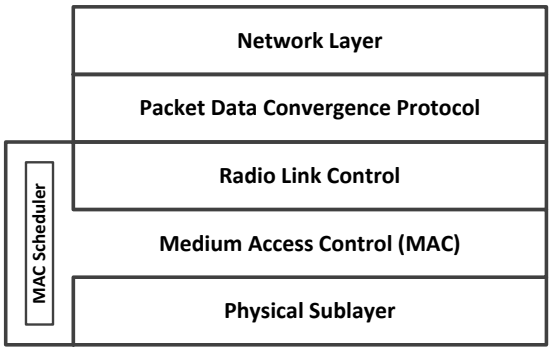


Figure 4.5: The 3GPP LTE E-UTRAN Data Plane Protocol Stack [17, 19, 30–32, 38, 97]

also responsible for providing ciphering functions to encrypt and protect all data traffic.

The RLC layer is responsible for the segmentation and concatenation of RLC SDUs received from the PDCP layer to create RLC PDUs. The size of each RLC PDU is determined by the MAC layer's scheduler, and forms part of the QoS control mechanism for the E-UTRAN. Depending on the sizes of each RLC SDU, and instructions issued by the scheduler, the RLC selects SDUs from the RLC's buffer and either segment them across multiple PDUs, or concatenate them to form larger PDUs. In addition to PDU construction, the RLC also provides an Automatic Repeat Request (ARQ) function to request the retransmission of any erroneous PDUs, and it ensures that all received packets are delivered to the upper layers of the stack in the same sequence that they were transmitted.

The MAC layer provides QoS management and radio resource control functionality within each cell. The RLC layer maps its PDUs onto a set of Logical channels which are managed by the MAC layer. Each Logical channel is defined according to the information it carries and whether such information is normal data traffic or control messages. There are five control and two traffic Logical channels in the downlink, and two control and one traffic Logical channel in the uplink. Control channels contain information such as paging, general system information, and RRC layer messages, while traffic channels simply contain mobile data which is exchanged between the Base Station and each Mobile Device.

Logical channels are multiplexed by the MAC layer onto Transport channels which specify how the information is to be transmitted at the Physical layer, as well

as specifying the information's characteristics. There are four Transport channels defined for the downlink, and two for the uplink. The Downlink Shared Channel (DL-SCH) and Uplink Shared Channel (UL-SCH) are the two primary Transport channels which provide dynamic rate adaption, scheduling in the time and frequency domains, Hybrid ARQ (HARQ), and spatial multiplexing for all of the traffic and most of the control Logical channels.

The primary responsibility of the MAC layer is to provide scheduling of radio resources to meet the QoS requirements for all traffic types. As is the case with 802.16, the scheduler is not defined in the LTE standard [17,30]. The MAC scheduler controls which RLC SDUs are encapsulated in RLC PDUs, as well as which RCL PDUs are multiplexed onto each Transport channel in the form of Transport blocks. The size of each Transport block, as well as the modulation scheme and antenna mapping performed at the physical layer, is determined by the MAC scheduler. In this way the scheduler is able to both allocate radio resources and perform adaptive modulation at the physical layer.

The scheduling of such resources is performed independently for both the uplink and the downlink. The Base Station's scheduler determines the radio resources to allocate to each Mobile Device, and the Mobile Device then decides how to allocate those resources between its Logical channels. This is in contrast to the 802.16 standard, in which all scheduling is performed by the Base Station and a Mobile Device is only able to request a redistribution of such resources.

The final task of the MAC layer is to provide HARQ for all Transport blocks. Although the RLC layer already contains an ARQ mechanism, it is significantly slower than the MAC's HARQ due to signalling feedback delays. The MAC ARQ on the other hand does not provide as reliable a retransmission function as the RLC ARQ due to its fast signalling feedback times. The combination therefore provides an extremely robust and reliable mechanism which is quick enough to be able to detect and retransmit erroneous packets without impacting on QoS [17].

Figure 4.6 presents the MAC header frame format for all MAC PDUs which are multiplexed onto Transport blocks. Each PDU contains a Logical Channel ID which enables the receiver to determine how to demultiplex the PDUs from each Transport block into separate Logical channels. The length of the MAC SDU or payload determines the number of bits used in the SDU Length field which is in turn determined by the Format bit. A 24-bit CRC value is also appended to each Transport block.

Figure 4.6: The 3GPP LTE MAC PDU Header Format for DL-SCH and UL-SCH [17, 19, 30–32, 38, 107]

4.2.2 The LTE Physical Layer

Each Transport channel defined by the LTE standard possesses its own variation of a physical layer which is linked to the QoS requirements of the type of traffic carried by the channel. Each such physical layer variation is referred to as a Physical Channel [110]. Figure 4.7 shows the primary physical layer functions for transmitting a transport block from the DL-SCH, referred to as the Physical Downlink Shared Channel (PDSC). The PDSC has been chosen as the focus for this section, as it contains most of the functions used in the various other Physical channels. The Physical Uplink Shared Channel (PUSC), which carries transport blocks originating from the UL-SCH, utilises a different multiple access scheme and includes an additional building block which performs a Discrete Fourier Transform (DFT) operation after the Symbol Mapper.

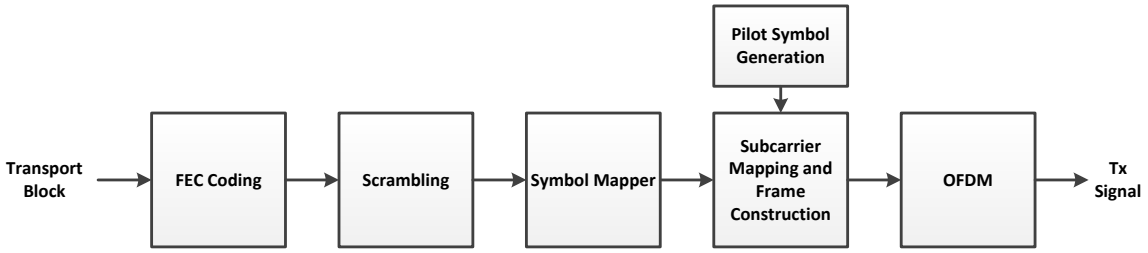


Figure 4.7: The 3GPP LTE Physical Downlink Shared Channel (PDSC) [17, 19, 30–32, 38, 108, 110]

The LTE standard mandates the use of Turbo Coding to perform FEC Coding in the PDSC [111]. The parameters for Turbo Coding are predefined for different Transport block sizes [111]. In the event that a Transport block's size is different to one of those listed in the standard, the Transport block must be segmented into

smaller blocks to conform to the standard. As the MAC scheduler defines the sizes of Transport blocks, it has indirect control over the coding rate and other parameters used by the Turbo coder, which means that it is able to indirectly perform adaptive coding based on channel conditions and QoS requirements.

The Scrambling stage of the PDSC performs a very similar function to that of the 802.16 standard's Randomiser; however the purpose of each function is very different. The 802.16 Randomiser is designed to provide physical layer encryption of transmitted signals, whereas the LTE Scrambling function reduces the interference resulting from signals being transmitted by neighbouring Base Stations by using a different scrambling sequence for each Base Station [111].

The LTE standard requires the use of the same symbol mapping, or digital modulation, schemes specified by the 802.16 standard, viz. QPSK, 16QAM, and 64QAM [110]. The modulation scheme used for each Transport block is defined by the MAC scheduler [107].

After symbol mapping has taken place, the modulated symbols must be assigned to specific frequency subcarriers and separate antennae. Depending on the application, the LTE standard supports transmit diversity, spatial multiplexing and beamforming schemes using multiple antennae. For reasons which are discussed in the following chapter, multiple antennae were not used to prove the viability of the CRRSA, and are considered beyond the scope of this thesis.

The mapping of Transport blocks onto specific frequency subcarriers is performed by the MAC scheduler [107]. A Resource block is defined by the LTE standard as the minimum number of subcarriers and time slots that the MAC scheduler may allocate to a Transport block. Each Resource block consists of 12 subcarriers of 15 kHz each, spaced over 6 or 7 OFDM symbols depending on the length of the cyclic prefix. In contrast to the 802.16 standard, which uses predefined Permutation patterns to specify the allocation of modulation symbols to subcarriers, the LTE MAC scheduler is free to use any method to divide the available subcarriers between Transport blocks. The LTE standard also specifies the use of pilot symbols, which are interspersed among the subcarriers carrying modulated symbols.

The LTE standard provides specifications for two types of Physical layer frames, depending on whether TDD or FDD is implemented [108]. Although it is likely that most LTE implementations will use FDD, this thesis utilises the TDD frame format for reasons which are provided in the next chapter. The LTE TDD Physical layer

frame format is presented in Figure 4.8, and consists of ten subframes divided into 0.5ms halves. The frame format is designed so that each subframe half matches the length of a Resource block, i.e. 6 or 7 OFDM symbols. The number of subcarriers used in the construction of the frame is dependent on spectrum availability. As is the case with the 802.16 standard, LTE was designed to accommodate flexible spectrum allocations, and thus the number of subcarriers can be varied according to the load carried by each Base Station.

Each subframe in Figure 4.8 can be used to transport either uplink or downlink signals depending on the demand for resource allocation in each direction. The three control fields in Subframes 1 and 6 provide guard intervals and information specifying the frame's uplink/downlink format.

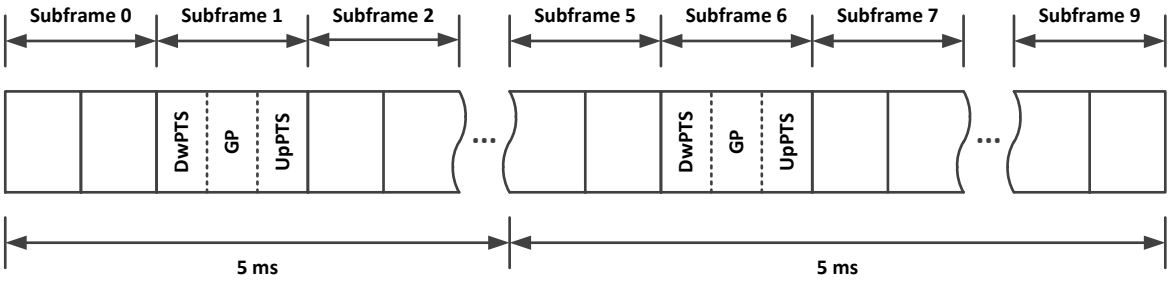


Figure 4.8: The 3GPP LTE Frame Structure Type 2 (TDD) [17, 19, 30–32, 38, 110]

Finally, after the Physical layer frame has been constructed, OFDM must be performed on each time slot in the frame. The length of the Cyclic Prefix (CP) used for each OFDM symbol depends on the channel conditions, and the standard provides two options, either of which can be implemented.

As was briefly mentioned at the start of the section, the LTE standard's PUSC implements a different multiple access scheme to the downlink channel (PDSC) [108]. The reason for this is that while OFDM is extremely spectrally efficient and is highly resilient when encountering ISI and ICI, it also generates signals which often have a high peak to average power ratio (PAPR). While the Base Station can transmit such signals without difficulty, the use of OFDM in the uplink would result in the Mobile Device draining its battery at an increased rate. The solution therefore is to adapt the traditional OFDM system by including a DFT stage prior to the subcarrier mapping, which spreads each Transport block across multiple subcarriers

and resource blocks prior to the standard OFDM operation. While not being as spectrally efficient as OFDM, this multiple access scheme, called Single Carrier FDMA (SC-FDMA), reduces the power consumption required to transmit signals, thereby increasing the battery lifespan of Mobile Devices. [17, 19, 30–32, 38]

4.3 Defining Reconfigurable Components

The previous two sections provided an overview of the IEEE 802.16 and 3GPP LTE RAT standards. The purpose of this section is to utilise that information to construct a set of Reconfigurable Components which contain the fundamental elements of each RAT which are required to establish and maintain a wireless link capable of successfully transmitting and receiving a signal carrying data without errors. Based on this objective, the functions selected from each RAT standard are those which are required to detect errors, those which provide the means to modulate data onto a signal, and those which assist in the successful transmission and reception of such a signal.

As discussed in Chapter 3, each Reconfigurable Component must contain a single atomic function which, wherever possible, is modelled so that it can be reused in the construction of different RATs. To achieve a high degree of reusability, Reconfigurable Components enable atomic functions to customise their functionality via the use of input parameters.

The decomposition of the 802.16 and LTE standards into their fundamental components results in the identification of both generic and RAT specific functions, which are listed in Table 4.1. It should be noted that even though some of these functions only occur in either 802.16 or LTE, they can be considered generic functions provided that they can be modified, via the use of parameters, to be capable of being employed in other RATs.

As previously indicated, both the 802.16 and LTE standards only provide specifications for the physical layer transmitter and not the physical layer receiver. For this reason they do not specify how to implement either Channel Estimation or Channel Equalisation. These two operations are critical however to the successful implementation of any RAT, and therefore two Reconfigurable Components have been developed to provide this functionality. The Channel Estimation Reconfigurable Component implements a Least Squares Channel Estimator which utilises

Table 4.1: Generic and RAT Specific Functions derived from the MAC and Physical Layers of the IEEE 802.16 and 3GPP LTE RAT Standards

Generic Functions common to both 802.16 and LTE	**802.16 Specific Functions**	**LTE Specific Functions**
• Calculate and Append CRC • Calculate CRC and Detect Errors • Convolutional Coding with Tail-biting / Decoding • Turbo Coding / Decoding • QPSK / De-QPSK • M-QAM / De-M-QAM • Packet Assembler / Disassembler • Subcarrier Mapping / De-mapping • OFDM / DeOFDM • Randomiser • Gold Randomiser	• Generate / Remove 802.16 Generic MAC Header • 802.16 Channel Interleaver	• Generate / Remove LTE MAC Header

Pilot symbols which are interspersed amongst the data subcarriers, and the Channel Equalisation Reconfigurable Component contains a Zero Forcing Equalisation algorithm [14, 16, 95, 98]. Although these Reconfigurable Components are not based on functionality contained in either the 802.16 or LTE standards, they can be still be used in the implementation of numerous different RATs, and are therefore examples of generic functions in the context of the RAT model.

As both the generic and RAT-specific functions encapsulated in Reconfigurable Components are specifically designed to be reused in numerous different RATs, each function supports the use of input parameters to determine how it should perform its operations. In a full CRRSA implementation of either the 802.16 or LTE standard, these input parameters would be provided by Reconfigurable Components containing dynamic link control, radio resource, and mobility management algorithms. In order to remain within the scope of this thesis, which is to provide two RATs which are capable of successful wireless transmission, these algorithms have been replaced with Reconfigurable Components which provide either statically defined or random parameters as appropriate. A list of these Reconfigurable Components, termed Abstracted Control Reconfigurable Components, is provided in Table 4.2 and the parameters for the 802.16 and LTE Abstracted Control Reconfigurable Components are provided in Chapter 5.

Table 4.2: Generic and RAT Specific Abstracted Control Reconfigurable Components

Generic Abstracted Control Reconfigurable Components	802.16 Abstracted Control Reconfigurable Components	LTE Abstracted Control Reconfigurable Components
• Generate Pilot Symbols • Generate Random Subcarrier Permutation	• WiMAX MAC PDU Control • WiMAX Adaptive Modulation and Coding • WiMAX Physical Layer Control • WiMAX Physical Frame Control	• LTE MAC PDU Control • LTE Adaptive Modulation and Coding • LTE Physical Layer Control • LTE Physical Frame Control

4.4 Modelling Radio Access Technologies

In order to demonstrate the CRRSA's ability to dynamically reconfigure and implement a RAT at runtime, this section describes the development of two RAT models derived from the 802.16 and LTE RAT standards respectively. These RATs are designed to establish and maintain a wireless link between the Subscriber and Access Network domains, capable of successfully modulating a Network layer packet onto a signal, transmitting that signal through the wireless environment, and successfully receiving and demodulating the signal without errors.

The RAT model derived from the 802.16 standard is presented in Figure 4.9, and the RAT model derived from the LTE standard is presented in Figure 4.10. Each of these diagrams illustrates and highlights which components are generic and which are RAT specific, and also which are Abstracted Control Reconfigurable Components. The data and control interfaces used to connect the Reconfigurable Components, and the logical separation between MAC and Physical layer functions, are also defined. To demonstrate that within the CRRSA there is no restriction in the Subscriber domain on the functionality implemented in Mobile Devices, the same RAT models are implemented in the Subscriber and Access Network domains.

Although the RAT policies used during the implementation are only described and discussed in Chapter 5, it is useful at this juncture to note that as some Reconfigurable Components may be reused in the same RAT model, it is necessary to assign each Component a unique ID number. This ID number is also used to define the connections between each of the Reconfigurable Components, and to ensure that the input and output specifications provided by each Reconfigurable Component match the connections in the RAT model.

In addition to satisfying the primary objective of using the RAT models to test the CRRSA's functionality, these models are also designed to demonstrate the flexibility and scalability which is generated by using Reconfigurable Components to create RATs:

- As the CRRSA does not place any limits on the number of input or output interfaces each Reconfigurable Component may contain, virtually any number of Reconfigurable Components connected in any topology could be used to construct a RAT model.
- The RAT Specific Reconfigurable Components which prepend MAC headers to each Network layer packet demonstrate that it is possible to both attach or inject signalling information into the transmission stream and extract it at the receiver. This capability is necessary to perform Radio Resource Control and other DLL operations.
- The "Generate Pilot Symbols" and "Generate Random Subcarrier Permutation" Reconfigurable Components show that it is possible for Reconfigurable Components to interact with Components in both the transmitter and receiver portions of the RAT model.
- In order to support dynamic operations such as Adaptive Modulation and Coding, Reconfigurable Components can use their dependent input functionality to provide a control channel which can be used by a scheduling algorithm to effectively turn specific Components on or off. This capability is also useful for routing packets through the RAT model. This is demonstrated by the Two Channel DeMUX Reconfigurable Component, which captures the output of the active modulation function.
- As data and control information is passed between Reconfigurable Components asynchronously, it is possible that related data and control information which is conveyed over different connections and via different Reconfigurable Components might not reach a particular Reconfigurable Component at exactly the same time. To prevent this, Reconfigurable Components are able to receive all related data and control information, alter only that which is relevant to performing the Reconfigurable Component's specific function, and transmit the result and all unaltered information to the next Component in the RAT model. This process is demonstrated by the Least Squares Channel Estimation Reconfigurable Component, which receives the Physical layer frame from the subcarrier, and transmits the unaltered frame along

with the Estimated Channel Transfer function to the Zero Forcing Equaliser Reconfigurable Component.

- The Channel Quality Measurement Reconfigurable Component demonstrates the Reconfigurable Component's inherent capability to request wireless environment measurements from the SDR, which can be processed and conveyed either to other Reconfigurable Components or, as is implemented in the RAT model, to the CRRSA's RMM for use by the RRRM in making reconfiguration decisions.

- Although not illustrated in either RAT model to avoid confusion, each Reconfigurable Component is able to submit context information to the CRRSA's RMM. In the case of the Reconfigurable Components used in the presented RAT models, such context information includes: the amount of time taken to process each set of inputs, the number of packets received, and the number of erroneous packets which are dropped.

The RAT Policies describing these RAT models and the parameters used to configure each of the Reconfigurable Component's functions are discussed in Chapter 5.

4.5 Summary

This chapter firstly provided an overview of two modern RAT standards the IEEE's 802.16 standard and the 3GPP's LTE standard. In order to demonstrate that the CRRSA is capable of implementing and reconfiguring RATs at runtime, two RAT models were developed based on the functionality and signal processing techniques contained in each of the chosen RAT standards. These RAT models are implemented in the following chapter, altogether with the CRRSA's required management system and operating environment.

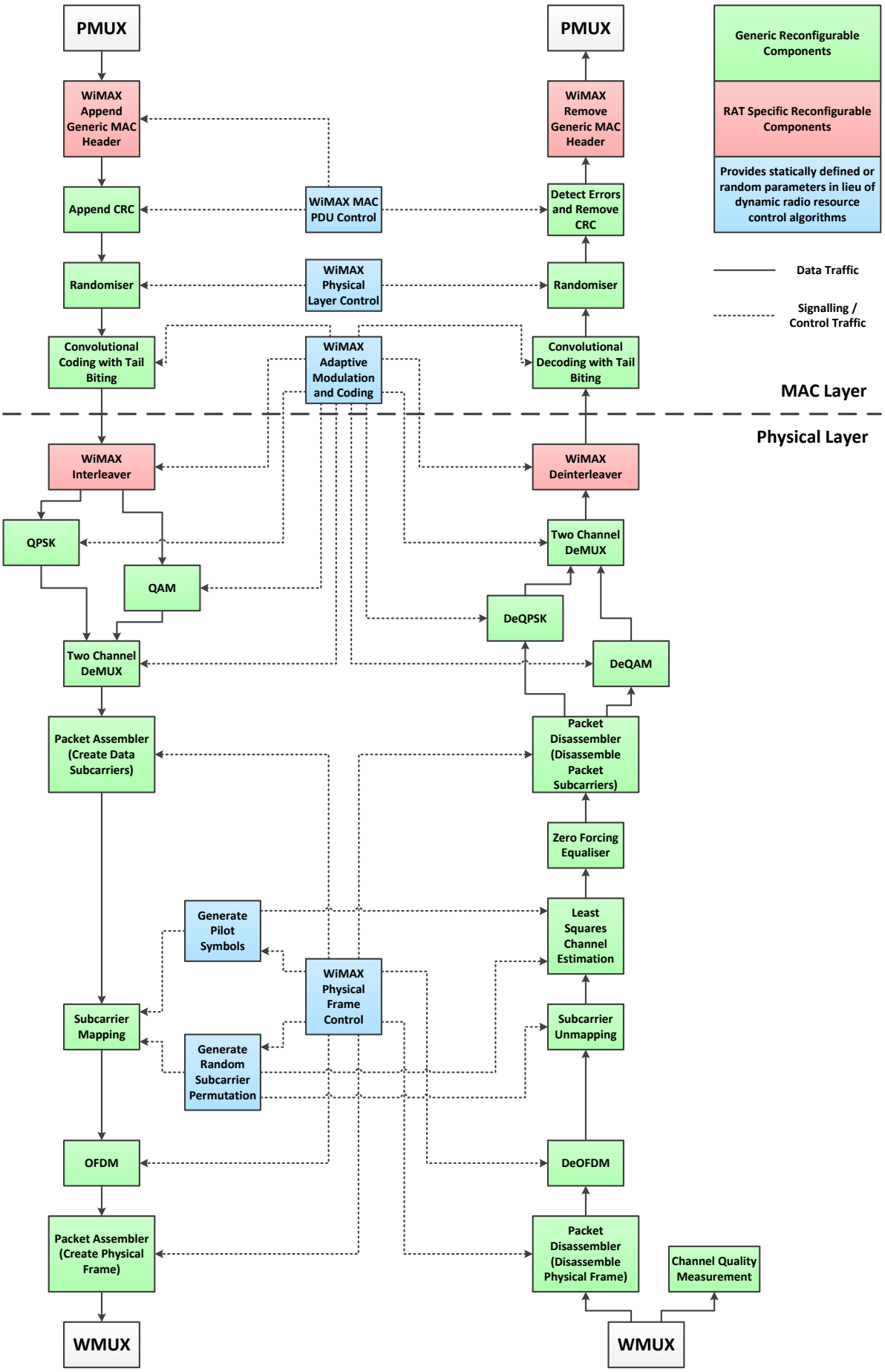


Figure 4.9: CRRSA RAT Model derived from IEEE 802.16

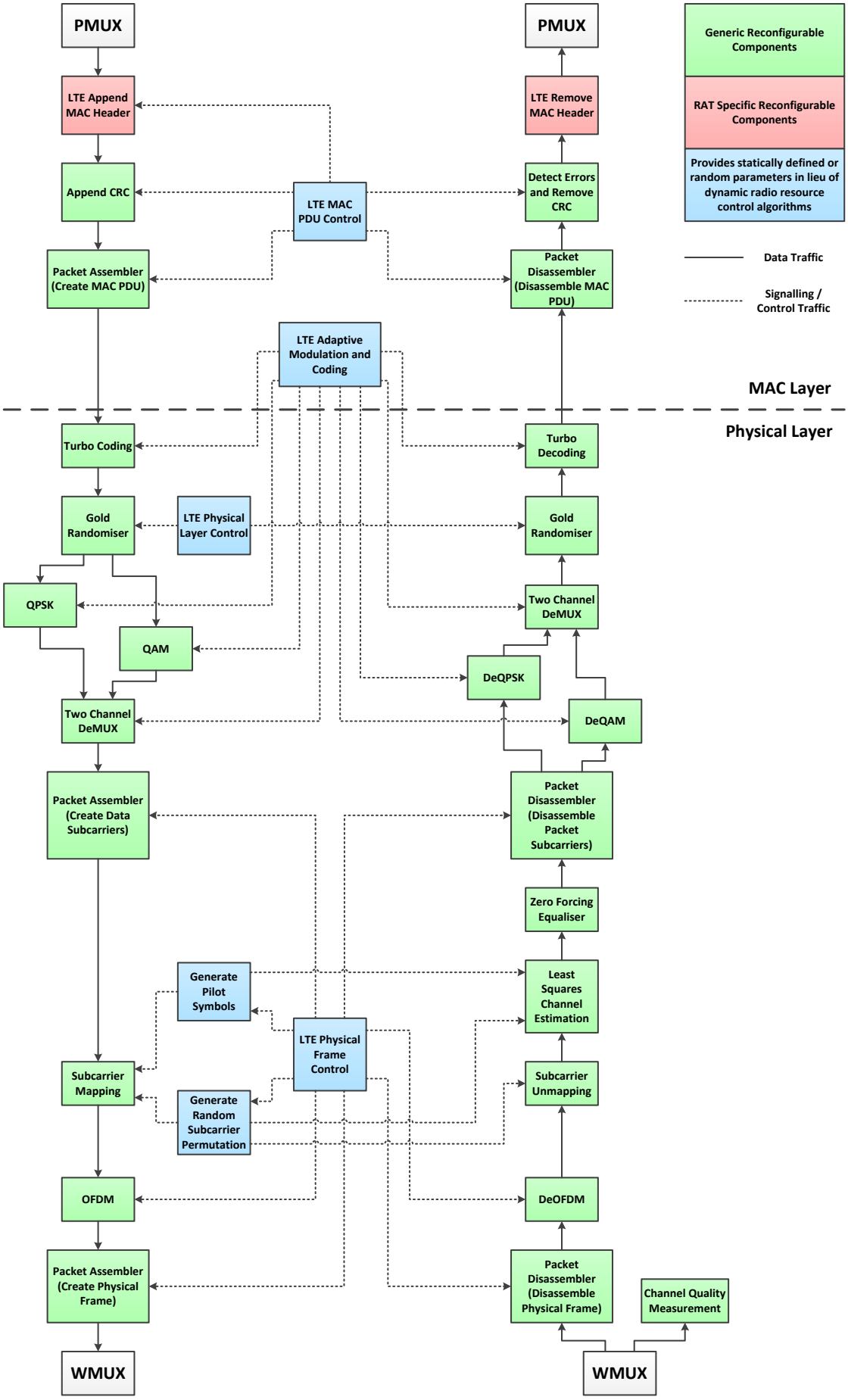


Figure 4.10: CRRSA RAT Model derived from 3GPP LTE

Chapter 5

Implementing the Converged Reconfigurable Radio Access Architecture

As outlined in Chapter 1, the objective of this thesis is to develop a RRS Architecture using a greenfield approach, which combines the functionality of all major RRSs whilst avoiding their individual limitations, and is capable of autonomously managing and controlling the intelligent and secure dynamic reconfiguration of multiple RATs within a RAN.

To accomplish this task, the work published by all major RRS research and standardisation initiatives was reviewed in Chapter 2, thereby resulting in the development of a set of functional requirements for the new Converged RRS Architecture. In Chapter 3 these functional requirements were encapsulated and implemented in the CRRSA, and its abstracted logical components, standardised interfaces and reconfiguration management operations were presented.

The purpose of this chapter is to demonstrate the practical viability of implementing the CRRSA, in doing so to prove that the CRRSA does meet all of its functional requirements, and to show that it is able to successfully transmit and receive a signal carrying data without errors while practically implementing the RAT models developed in Chapter 4.

The rest of the chapter proceeds as follows: Section 5.3 defines the objectives of the implementation, Section 5.2 presents each of the supporting technologies which are used in the implementation, Section 5.3 discusses how the CRRSA's logical components, interfaces, and reconfiguration management operations have been im-

plemented and tested to demonstrate the viability of the CRRSA, and Section 5.4 presents and critically analyses the outcomes and results of the implementation.

5.1 Implementation Objectives

To verify that the CRRSA is capable of performing all necessary reconfiguration management operations, the implementation thereof must demonstrate the following functionality:

- The reconfigurable management systems in each of the domains can be independently and autonomously booted up and are then able to initialise and execute their operations using parameters contained within predefined Network Equipment and RAT policies.
- During initialisation operations in the Subscriber and Access Network domains, the default RAT(s) are implemented and used to establish and maintain a wireless connection between network elements in the Subscriber and Access Network domains.
- Reconfigurable Algorithms can be dynamically instantiated and their parameters changed at runtime, in order to reconfigure each network element's radio resource and traffic distribution, its spectrum allocation and transmission parameters, as well as its secure inter-domain communication protocols.
- RATs can be created or terminated at runtime based on decisions which rely on monitoring the status and performance of all RATs and comparing that information with predefined network equipment parameters and restrictions, and after considering a list of available RATs together with their specific performance metrics.
- New Policies, as well as Reconfigurable Components and Algorithms, are automatically uploaded from the Core Network to network elements in the Subscriber and Access Network domains.

In addition to the reconfiguration management operations discussed above, the CRRSA implementation must also verify that:

- Multiple different RATs can be successfully created and run simultaneously using a variety of Reconfigurable Components as well as information contained in RAT policies.
- RATs must be able to modulate Network Layer packets or RRS management messages onto a signal, transmit that signal through the wireless environment using SDR hardware platforms, and demodulate the received signal and retrieve the original packet or message without errors.

5.2 Implementation Technologies

A number of software and hardware technologies were employed during the implementation of the CRRSA. The following subsections provide an overview and the basis for selecting these technologies to implement the CRRSA.

5.2.1 The Java Programming Language and the Java Agent Development Framework (JADE)

Java is an object-oriented programming language which is widely used to develop PC, mobile and web-based applications. It provides all the standard functionality required of an object-oriented programming language, including inheritance, polymorphism, abstraction, encapsulation, method overloading and multi-threading. A key feature of Java is the ability of its applications to be deployed on virtually any hardware platform which is capable of running the Java Virtual Machine (JVM). (See [112])

The CRRSA implementation was developed using Java as the principal programming language. The primary motivating factor for choosing Java was that it is capable of compiling and dynamically incorporating classes into the runtime environment [77, 112, 113]. This capability is used to compile and instantiate Reconfigurable Components and Algorithms in the Subscriber and Access Network domains at runtime, after they have been downloaded from the Core Network. This is turn demonstrates the CRRSA's ability to download new Reconfigurable Components and Algorithms and use them to execute at runtime reconfiguration operations such as creating a RAT.

The Java Agent Development Framework (JADE) is a software platform and development framework that enables the creation of software agents in Java [114, 115]. A software agent is a component of software which is capable of autonomous operation as well as communication with other agents according to a predefined communications protocol [115]. Because agents are capable of independent and autonomous operation, they offer the ideal mechanism for practically implementing the CRRSA's logical components. The CRRSA's standardised interfaces for exchanging information between logical components are implemented using peer-to-peer communications between agents. In doing so, the use of JADE agents demonstrates that the CRRSA adheres to the criteria defined in Chapter 1, with particular regard to the need for the CRRSA's components to be logically defined and distributed across domains and for the interfaces between such components to be well-defined.

JADE was first released by Telecom Italia in 1998 as a means to test the IEEE's Foundation for Intelligent Physical Agents (FIPA) standard, and as such it implements much of the functionality specified in the FIPA standard [115]. All JADE agents are executed on the JADE platform, which provides all the functionality for launching, hosting, and managing agents. Figure 5.1 provides an illustration of the JADE architecture and its logical components and operations.

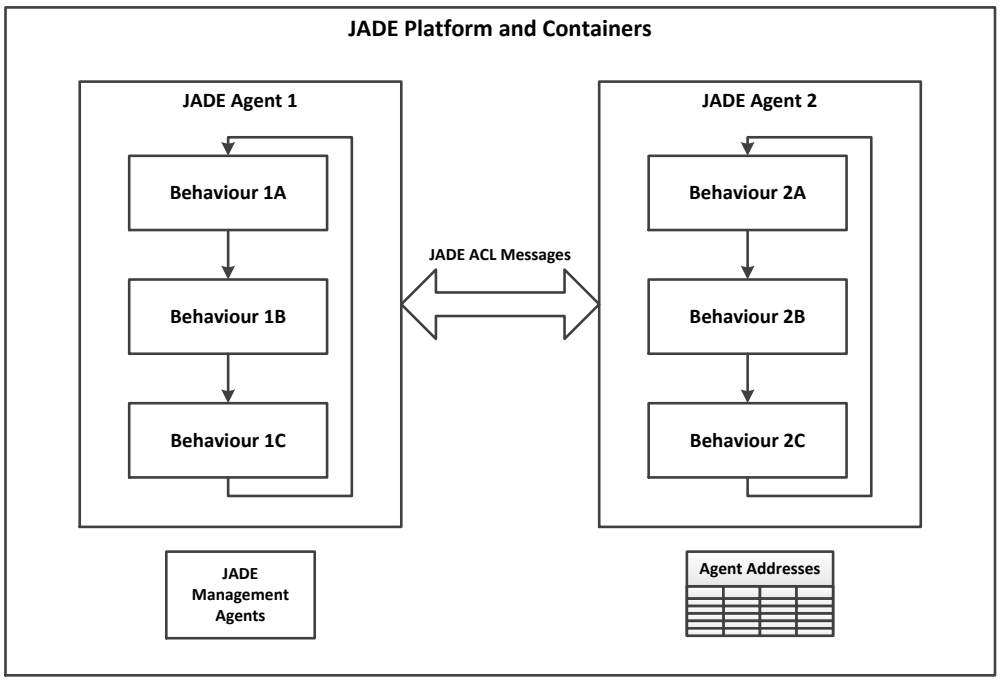


Figure 5.1: The JADE Architecture (Adapted from [115])

JADE agents are organised into containers and assigned unique addresses by JADE. Each JADE agent runs in a separate thread and is capable of communicating with any other agent using standardised Agent Communication Language (ACL) messages. Each ACL Message contains a performative to identify the nature of the message (e.g. request, reply, or inform), the addresses of all the JADE agents to which the message is being sent, the address of the JADE agent which has created and sent the message, a conversation identifier to differentiate between messages passed between the same set of agents, and the content of the message. Agents buffer any messages they receive until it is possible to process them [115].

The functions performed by a JADE agent are implemented in a set of classes called Behaviours. When a JADE agent is launched, instances of specified Behaviour classes are instantiated and then run serially as shown in Figure 5.1. Depending on the nature of the functionality being implemented, there are several different types of Behaviours available [115]. Each type of Behaviour differs in terms of how often it is executed, for example a Cyclic Behaviour is always be executed, whereas the Ticker Behaviour is skipped until a predefined amount of time has passed [115]. In the event that multiple Behaviours are required to be executed simultaneously, JADE provides the functionality needed to create additional threads for specific Behaviours within each agent [115].

To prevent processing resources being wasted on executing Behaviours which are waiting for messages, a JADE agent provides the ability to place a Behaviour into a blocking state. In this state the Behaviour is not be executed until a message has been received by the JADE agent [115].

5.2.2 The MATLAB Programming Language and Development Environment

The Matrix Laboratory, or MATLAB as it is more commonly known, is a functional and object-oriented programming language and development environment, developed by MathWorks and widely used by both academia and industry to develop and implement complex mathematical algorithms. The MATLAB development environment provides a number of libraries, or toolboxes, which contain commonly-used and standardised mathematical functions. Amongst these toolboxes are the Communications System Toolbox [116] and the Signal Processing Toolbox [117], which together contain most of the digital signal processing techniques needed to implement the Reconfigurable Components developed in Chapter 4 and the Reconfigurable WMUX Algorithm presented in Section 5.3.2.3. below.

In order to utilise the functionality provided by MATLAB in the CRRSA implementation, a method was required to enable MATLAB Functions to be able to be executed in Java applications. The two methods identified and utilised are depicted in Figure 5.2.

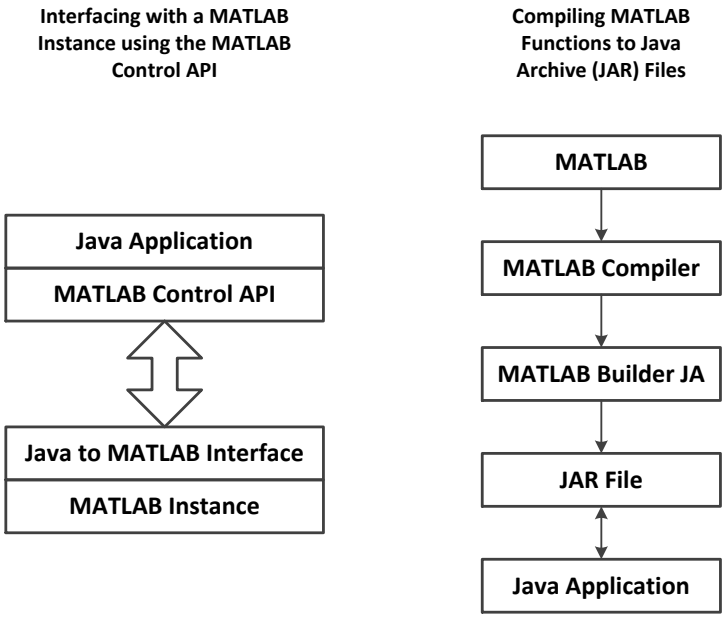


Figure 5.2: The MATLAB Control API and MATLAB Builder JA Compiler used to execute MATLAB Functions in Java applications

The MATLAB Builder JA compiler [118] enables MATLAB Functions to be compiled to Java Archive (JAR) files which can be executed by Java applications. The advantage of using the MATLAB Builder JA compiler is that it allows the resulting JAR files to be deployed on any platform provided that the Java MATLAB library has been previously installed on that platform. There are however a number of MATLAB functions which cannot be compiled using MATLAB Builder JA compiler and to accommodate these functions an alternative solution has to be identified.

The MATLAB Control API [119] is an open source software project which is capable of interfacing a Java application with an instance of MATLAB, thereby enabling it to both exchange data with and issue commands to the MATLAB instance. Although the MATLAB Control API does enable a Java application to utilise any of the functionality provided by MATLAB and its toolboxes, it does require that an instance of MATLAB is running during the execution of the Java application. In

the CRRSA implementation this presents a problem, as using the MATLAB Control API to implement all Reconfigurable Components and the Reconfigurable WMUX Algorithm would require that they all use a single instance of MATLAB, thereby removing the performance gains achieved by the use of multi-threading. This impact of this problem is mitigated by using a combination of the MATLAB Builder JA compiler and the MATLAB Control API, with functions which are unable to be compiled by the MATLAB Builder JA compiler implemented using the MATLAB Control API.

5.2.3 The Universal Serial Radio Peripheral (USRP) and the USRP Hardware Driver (UHD)

The Universal Serial Radio Peripheral (USRP) is a relatively inexpensive SDR hardware platform manufactured by Ettus Research and widely used by academia and SDR researchers [120]. The USRP was originally developed as a hardware platform to use in combination with communication system simulations developed in GnuRadio [82]. Ettus Research has since developed a library called the USRP Hardware Driver (UHD), which allows USRP radio applications to be developed in C++ [121].

The USRP model used in the CRRSA implementation is the USRP2, which is capable of transmitting and receiving RF signals with a maximum bandwidth of 50MHz. Each USRP2 contains a motherboard which performs the baseband digital signal processing, and a daughterboard which modulates the baseband signal onto a specific carrier frequency. There are a number of different daughterboards available for use with the USRP2, each of which is capable of transmitting and receiving signals at different frequencies.

As illustrated in Figure 5.3, the CRRSA implementation uses two USRP2s, which together represent network elements in the Subscriber and Access Network domains (i.e. a Mobile Device and a Base Station) and which are connected to a PC via Gigabit Ethernet cables. Each USRP2 is assigned an IP address which is used to establish communications between the PC and the USRP2. The USRP2 is also capable of supporting multiple antennae; however this requires a separate USRP2 for each antenna, and these would then be linked using a timing and frequency synchronisation cable. As previously mentioned, the implementation of multiple antennae signal processing techniques such as spatial multiplexing is beyond the scope of this thesis.

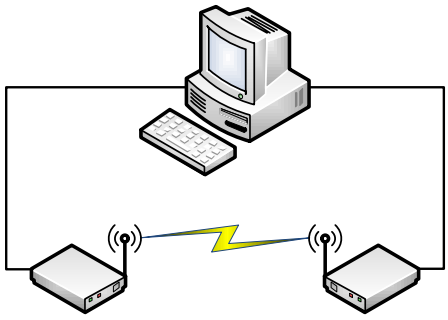


Figure 5.3: The USRP2 Setup and Layout

The CCRA implementation running on the PC contains a special C++ SDR API wrapper class which enables the CRRSA's SDR API logical component written in Java to interface with the UHD written in C++. The C++ SDR API wrapper class enables the CRRSA implementation to connect to each USRP2, and to transmit and receive RF signals, set the gain, carrier frequency, and sampling rate parameters, and to measure the Received Signal Strength Indicator (RSSI) value on all incoming transmissions.

The daughterboard used in the CRRSA implementation is the XCVR2450, which was chosen as it is capable of transmitting and receiving signals in the ISM bands at 2.4GHz and 5.9GHz. The disadvantage of using the XCVR2450 is that it can only perform half-duplex communications, which forces the CRRSA implementation to utilise Time Division Duplexing (TDD). (See [122])

The synchronisation of timing and frequencies between the USRP2s is important in order to ensure reliable error-free transmission. In conventional mobile telecommunication systems an external synchronisation signal is employed to align the clock frequencies of all network elements. This external synchronisation signal could be implemented using the Global Positioning System (GPS) or via a general clock signal originating in the core network. To synchronise the two USRP2s used in the CRRSA implementation, the multiple antenna synchronisation cable is used to align each radio's internal clock. This method is not perfect however, which is why the Reconfigurable WMUX Algorithm is required to synchronise each received signal.

5.2.4 Extensible Markup Language (XML)

Extensible Markup Language (XML) is a markup language standardised by the World Wide Web Consortium (W3C) [123] that enables information to be captured and stored in a document according to a predefined format and structure which is both human and machine readable. Information is stored and identified in an XML document via the use of markup elements and tags, which are customised to suit a particular application and the information stored in the document [124]. Many software applications utilise XML to store data structures or other information in a platform-independent predefined format which can be transported between platforms and/or used by other applications.

The CRRSA's Network Equipment and RAT policies are implemented in XML documents according to a specific format and structure, which is described in Section 5.3.3. XML is well suited to defining these policies, as it allows the format and structure of each policy to be defined, but does not place any restrictions on the content.

The Document Type Definition (DTD) is a schema language for XML, and is used to define the format, structure, elements, and tags which must be used when creating a specific XML document [124]. To validate Network Equipment and RAT policies, the CRRSA implementation provides DTD documents for each type of policy. These DTD files are used by each domain's RPM to determine whether each policy contains the required information and also to detect if it has been corrupted during transmission between domains.

There are a number of programming interfaces which have been developed to create or extract information from XML documents. The Simple API for XML (SAX) is one such interface which allows the contents of an XML document to be serially extracted and saved in predefined data structures for use in software applications. The CRRSA implementation uses the SAX parser library for Java [124] to read each Network Equipment and RAT policy and store the information in an instance of a Java class written for the purpose. Section 5.3.3 provides more information regarding the specific Network Equipment and RAT policies used in the CRRSA implementation.

5.3 Implementing and Testing the Converged Reconfigurable Radio System Architecture

The previous sections have outlined the objectives and technologies used in the implementation, and have provided the context for the discussion concerning specific implementation issues which is presented in the following subsections.

5.3.1 Overview of the Implementation and the Simulation Environment

The simulation environment provides a framework for the CRRSA's logical components, interfaces, and operations to be implemented and tested using different parameters and policies in all three separate wireless environment scenarios. Figure 5.4 presents an overview of the simulation environment and the CRRSA implementation.

As previously mentioned, all logical components in the CRRSA implementation have been implemented in the form of JADE agents. These agents are organised into three separate containers which represent the Subscriber, Access Network, and Core Network domains. This separation does not prevent agents in one container from communicating with those in another container; it merely provides a logical separation to highlight the fact that different logical components in the CRRSA are located in separate domains.

The Network Equipment and RAT XML policy files and the Reconfigurable Component and Algorithm Java class files are stored in separate folders for each domain, representing the Reconfigurable Policy Database and the Reconfigurable Component and Algorithms Database which are contained in each domain.

Figure 5.5 presents the operations which are performed during the initialisation of each simulation. The parameters for each simulation are entered by the user via either a software wizard which guides the user through the various simulation options, or a command line interface, which is better suited to setting up a series of simulations. The Simulation Controller is a JADE agent responsible for storing the simulation parameters, and for making them accessible to the relevant JADE agents after they have been launched.

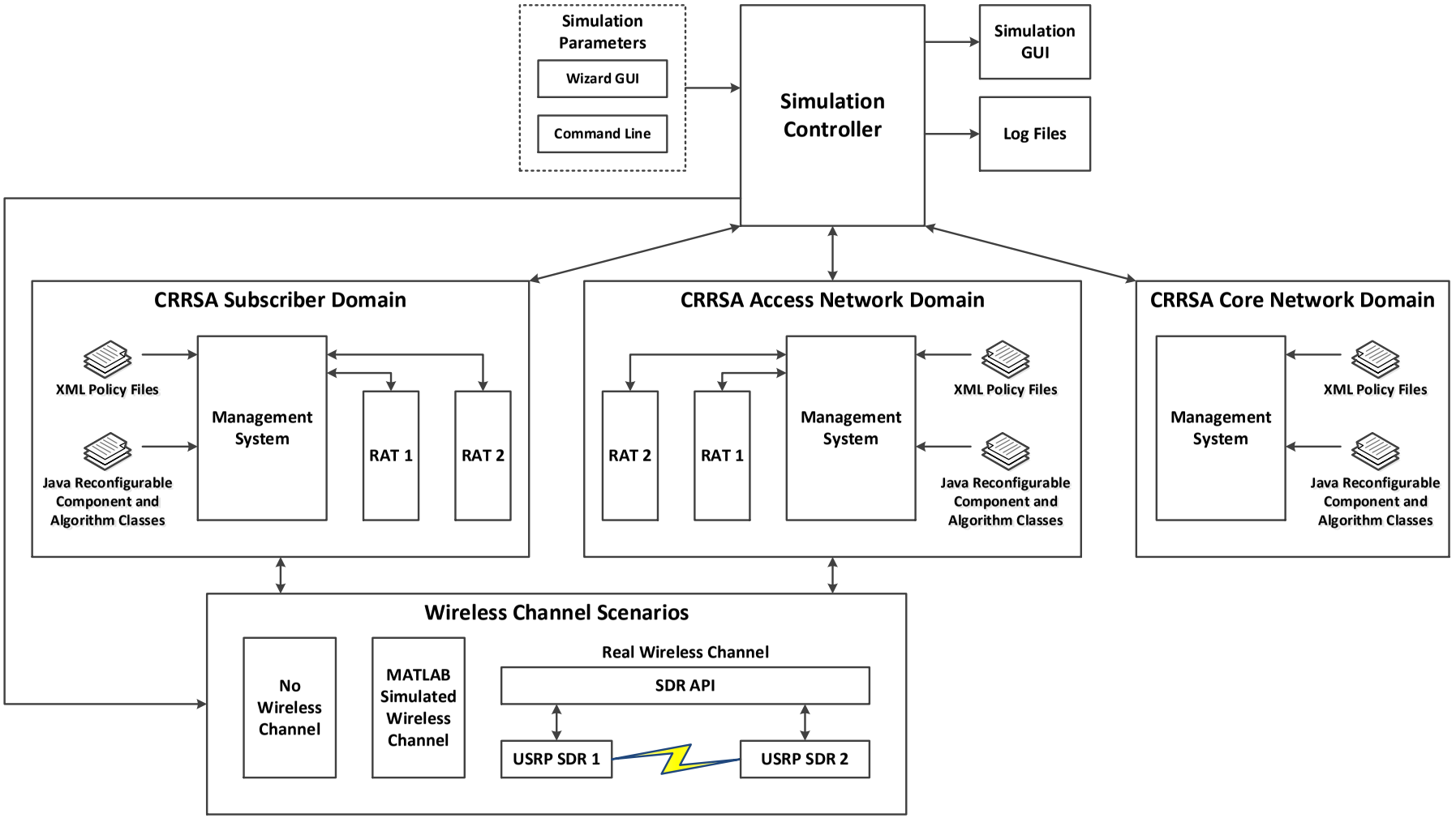


Figure 5.4: Overview of the Simulation Environment and CRRSA Implementation

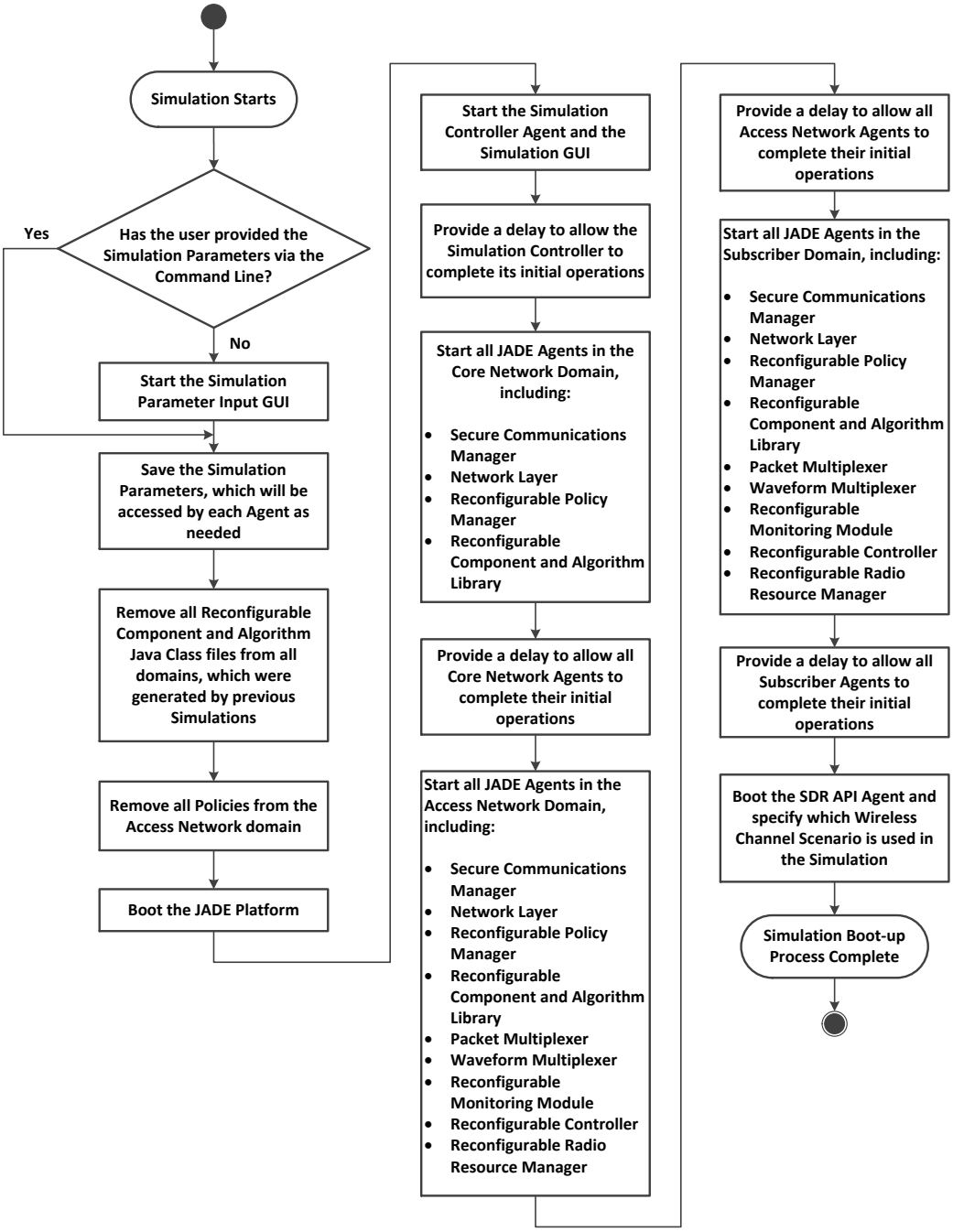


Figure 5.5: Initialising the Simulation Environment and CRRSA Implementation

During the simulation initialisation process, the Policies and Reconfigurable Components and Algorithms contained in the Access Network domain's database are deleted, which forces the Access Network domain's RPM and RCAL to request new files from the Core Network. This demonstrates the CRRSA's ability to download and install policies and software components, thereby fulfilling one of the implementation objectives. This process is not performed on the files in the Subscriber domain, as a Mobile Device must first implement a RAT before it can establish a connection to the Core Network, therefore all the necessary Policies, and the Reconfigurable Components and Algorithms, must already be present in the Subscriber domain's databases at the start of the simulation.

During a simulation the RCALs in each of the domains produce Java class files whenever they compile and instantiate Reconfigurable Component and Algorithm classes. To ensure that each simulation starts from a predefined state, and also to demonstrate the functionality provided by each RCAL, these files are always deleted at the start of a simulation.

To monitor the progress of a simulation, the Simulation Controller receives status reports from each logical component, as well as performance measurements for each of the RAT implementations. This information is presented on a Graphical User Interface (GUI) and is also optionally stored in plain-text log files. The simulation environment provides the general support mechanisms required to run each simulation and to monitor its progress; however to demonstrate the CRRSA's reconfigurability functionality, and thereby meet the implementation objectives described in Section 5.1, a specific simulation scenario is required.

This simulation scenario is formulated through the implementation of Network Equipment and RAT policies, and Reconfigurable Components and Algorithms. Perhaps the most important part of this simulation scenario involves the implementation of the Reconfigurable Decision Making Algorithm, as it defines how the reconfiguration operations are performed in the Subscriber and Access Network domains. The details for this Algorithm, the other software components, and the Network Equipment and RAT policies used in the implementation, are provided in the following sections.

In addition to testing the CRRSA's reconfigurable management system, it is also necessary to demonstrate that the CRRSA is capable of dynamically implementing RATs at runtime, and that such RATs are able to transmit a signal carrying Network Layer packets though a wireless channel without errors. These Network Layer

packets either contain RRS Management messages from a SCM or, in the absence of such messages, each packet's contents are randomly generated.

To determine whether each of the implemented RATs is capable of successful wireless transmission, the simulation environment provides three different Wireless Channel Scenarios. Each Wireless Channel Scenario provides different testing conditions which are used to demonstrate different aspects of the CRRSA's functionality:

- **The No Wireless Channel Scenario**: is used to demonstrate that each implemented RAT functions correctly, and is able to modulate a Network Layer packet onto a signal, and then demodulate that signal to retrieve the original packet. An additional benefit of using this scenario is that it abstracts the operations carried out by the CRRSA's reconfigurable management system from the performance of each implemented RAT.
- **The MATLAB Simulated Wireless Channel Scenario**: provides the opportunity to test the performance of each RAT implementation in a simulated wireless environment. This scenario is implemented using the Additive White Gaussian Noise (AWGN) and Rayleigh Fading Channel functions provided by the MATLAB Communications Systems Toolbox. At the start of each simulation, the user is able to define the Signal-to-Noise (SNR), Signal Power, Carrier Frequency, and Delay Profile used in the simulation.
- **The Real Wireless Channel Scenario**: provides the means to test whether each RAT is capable of successfully transmitting a signal through the wireless environment without errors. This scenario is implemented using a combination of the USRP2s introduced in Section 5.2, and the SDR API defined by the CRRSA. By demonstrating that the CRRSA implementation is capable of transmitting a signal between the Subscriber and Access Network domains using this scenario, it proves that the CRRSA is practically realisable and therefore meets one of the core objectives of this thesis.

The simulation environment and the CRRSA implementation constitute an event-driven simulation, which means that components in the simulation only perform their tasks when specific events occur. Although this type of simulation is both necessary and standard practice for implementing an abstracted component-based model such as the CRRSA, the speed with which each event-triggered operation is carried out is determined entirely by the hardware running the simulation. As the purpose of the CRRSA is to demonstrate functionality and not to measure

performance this issue is not a problem, however it must be borne in mind when running the simulation and/or examining its results.

The CRRSA Implementation, which includes all policy files, MATLAB functions, the source code for the SDR API C++ wrapper, all Java source code, and the compiled binaries to run the simulation, has been stored on a CD attached to this Thesis. A guide to the CD is provided in Appendix A.

5.3.2 Implementing Reconfigurable Algorithms

To ensure that developers are provided with a standardised API for creating new Reconfigurable Algorithms, the CRRSA implementation defines root classes with abstract methods for each type of Reconfigurable Algorithm. Each Reconfigurable Algorithm is implemented as a subclass and overrides the specific abstract methods.

The CRRSA implementation utilises a naming convention for each type of Reconfigurable Algorithm, to enable the RCAL to differentiate between the Reconfigurable Components and each type of Reconfigurable Algorithm in its database. This naming convention is very simple, requiring all file names to start with one of the following letter formats, which relate to the type of Reconfigurable Algorithm: DM (Decision Making), PMUX, WMUX, and CA (Communications Algorithm).

After the RCAL has compiled the Algorithms at runtime, the RRRM must inform each relevant logical component of the name of its Reconfigurable Algorithm's class and the input parameters to be used during initialisation. The Java Reflection API is then used to perform the instantiation. The following subsections outline the Reconfigurable Algorithms developed to demonstrate the CRRSA's functionality.

5.3.2.1 Implementing a Reconfigurable Communications Algorithm

The Reconfigurable Communications Algorithm used in the CRRSA implementation is based on a Stop and Wait ARQ algorithm. Although this algorithm is the simplest ARQ method available, it provides the functionality needed to ensure that reconfigurable management messages are successfully exchanged between domains, even if some packets are lost in transit due to the effects of the wireless environment [95, 125, 126].

The Stop and Wait ARQ algorithm's method of operation, illustrated in Figure 5.6, is fairly simple and works as follows [95, 125, 126]:

1. A packet is transmitted from SCM A to SCM B, whereupon SCM B transmits an acknowledgement message (ACK) back to SCM A.
2. If a message is lost during transit, SCM A does not receive an ACK from SCM B within a predefined period of time, and so SCM A retransmits the message.
3. If the ACK is lost during transit, SCM A assumes that its packet did not reach SCM B within a predefined period of time, and so SCM A retransmits the message.
4. If packets are repeatedly lost during transit, both SCMs eventually determine that communication is not possible and any further attempts at transmitting the packets are abandoned for the time being.

Whenever an SCM receives a reconfigurable management system message which must be transmitted to another SCM in a separate domain, it splits the message into separate packets and prepends each packet with a header specifying the source and destination domains, the number of packets in the transmission series, and whether a packet has any padding. Figure 5.7 presents the packet format used in the Stop and Wait ARQ Reconfigurable Communications Algorithm.

To determine whether a packet has been corrupted during transit, the Stop and Wait ARQ Reconfigurable Communications Algorithm appends a CRC value to each packet. If the CRC value calculated at the receiver does not match the CRC appended to the packet at the transmitter, the packet is considered to be corrupted and no ACK is sent to the transmitting SCM.

The Stop and Wait ARQ Reconfigurable Communications Algorithm defines a set of input parameters which determine the size of each packet's payload, the duration of the transmit and receive timers, and how many times the transmit timer can expire before the transmission attempt is abandoned. Table 5.1 contains the input parameter values used in the CRRSA implementation. The choice of values is determined by how long it could take for a single packet to be transmitted and received. Although these values may seem excessive they are necessary as the processing resources used by simulation are shared between numerous agents, resulting in slower processing times for all.

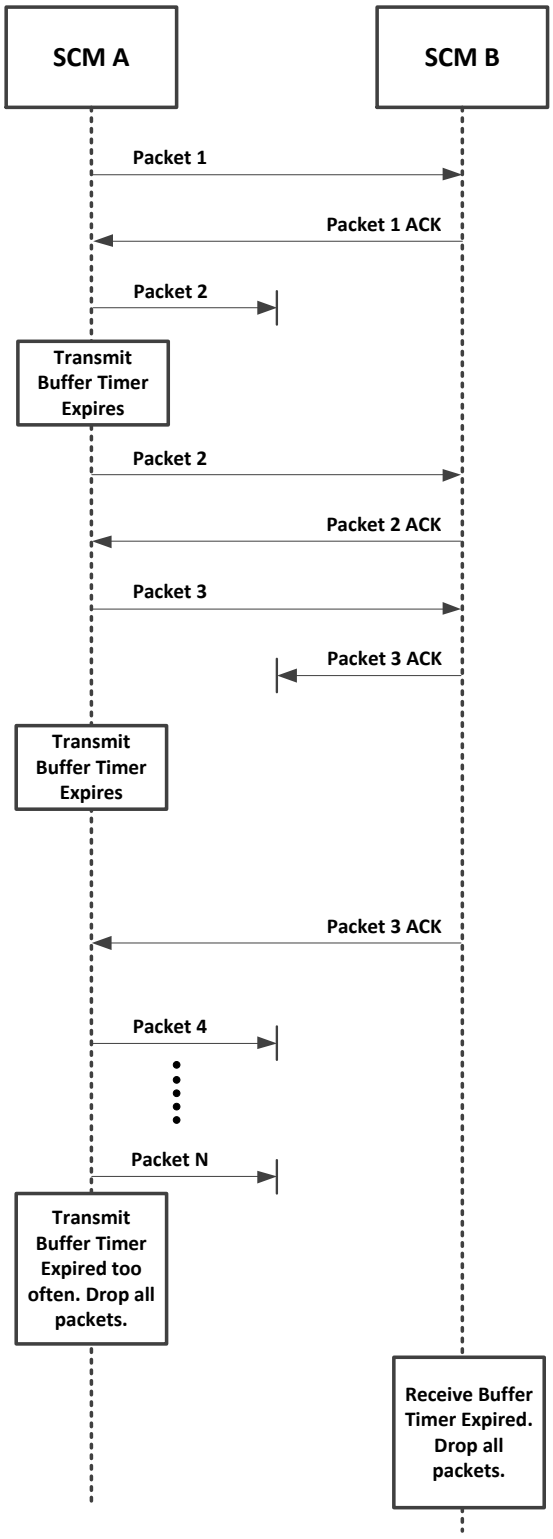


Figure 5.6: The Stop and Wait ARQ Algorithm (Derived from [95, 125, 126])

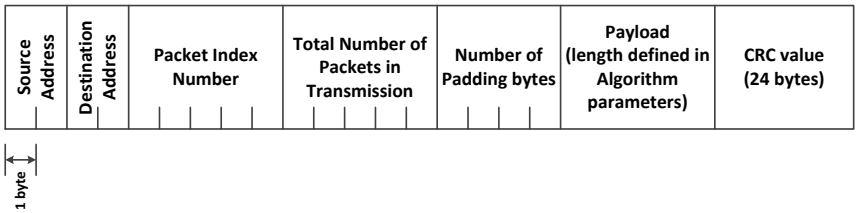


Figure 5.7: The Stop and Wait ARQ Reconfigurable Communications Algorithm's packet format

Table 5.1: The Stop and Wait Reconfigurable Communications Algorithm's input parameters for the CRRSA implementation

Parameter Name	Parameter Value
Packet payload length	12 bytes
Transmit timer duration	10 minutes
Number of times the transmit timer can expire before transmission considered a failure	100
Receive timer duration	20 minutes

Figure 5.8 provides a detailed description of the operations carried out by the Stop and Wait ARQ Reconfigurable Communications Algorithm, which can be combined with the internal operations description provided in Chapter 3 to produce a complete representation of how the SCM was implemented in the CRRSA implementation.

5.3.2.2 Implementing a Reconfigurable Packet Multiplexing Algorithm

The Reconfigurable Packet Multiplexing (PMUX) Algorithm manages the traffic scheduling between RATs based on parameters provided by the RRRM. The Reconfigurable PMUX Algorithm's implementation determines which Network Layer packets should be routed to different RATs based on the contents of each packet's header, and is therefore called the Header Routing Reconfigurable PMUX Algorithm.

There are two types of Network Layer packets produced in the CRRSA implementation, namely reconfigurable management system messages and random Network Layer packets. The random Network Layer packets are used to test the functionality of each implemented RAT, and their header format cannot be predefined. Thus the PMUX is only able to differentiate between reconfigurable management system messages and random packets. Although this may seem like a limitation, the CRRSA

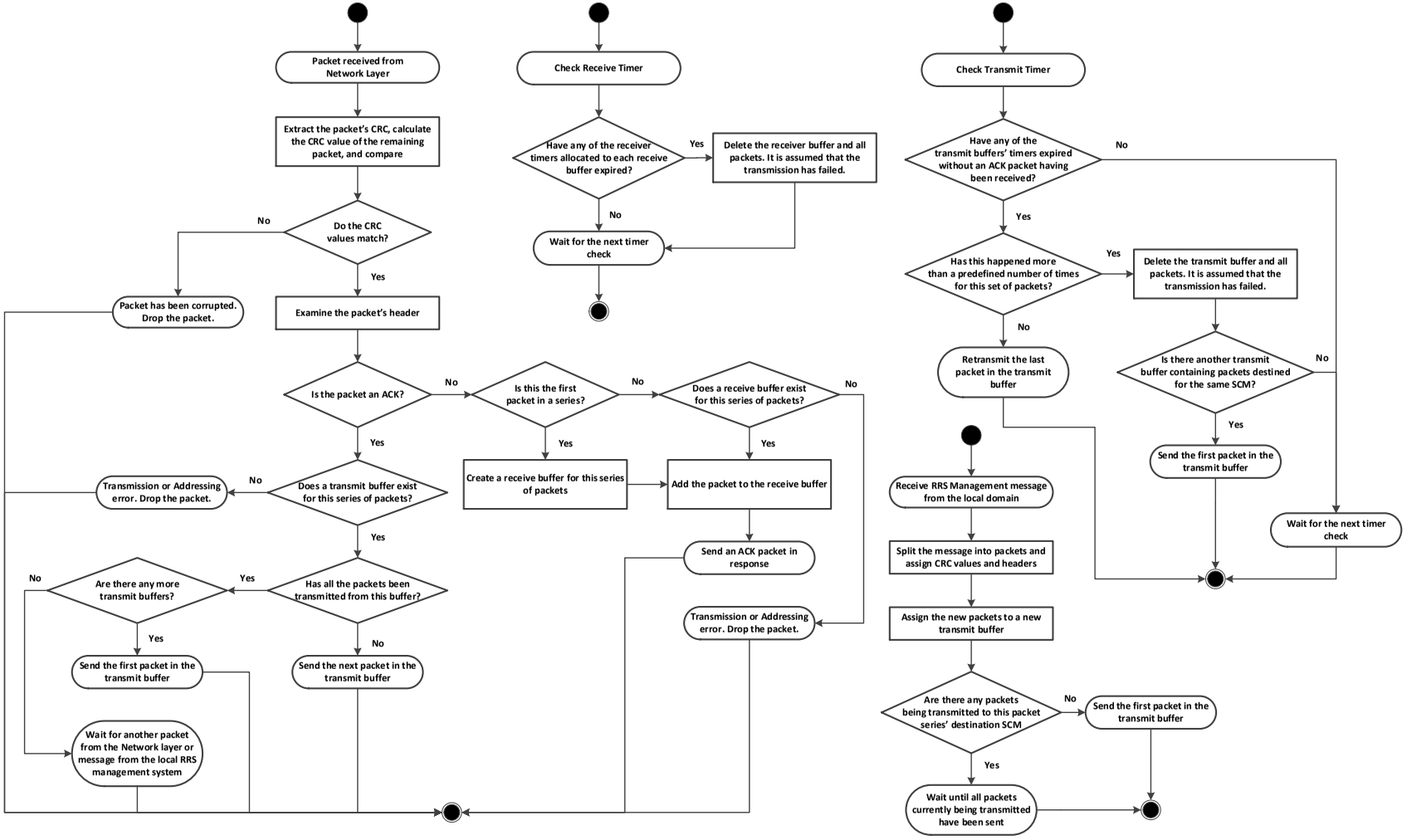


Figure 5.8: The operations performed by the Stop and Wait ARQ Reconfigurable Communications Algorithm (Derived from [95,125,126])

implementation only needs to demonstrate that it is capable of scheduling traffic between RATs, which is accomplished using the Header Routing Algorithm.

As each RAT is created by the RC, the addresses of the RAT's Reconfigurable Components are sent to the RRRM. The RRRM then uses this information to instruct the Header Routing Algorithm to route reconfigurable management system messages to specific Reconfigurable Components in each RAT. The Header Routing Reconfigurable PMUX Algorithm's parameters are primarily defined by the RRRM at runtime; however the Network Equipment policies do specify the length of the Packet header to be used by the Algorithm to determine where to route the packet. The specific input parameters and values used in the CRRSA implementation are defined in Table 5.2.

Table 5.2: The Header Routing Reconfigurable Packet Multiplexing Algorithm's input parameters for the CRRSA implementation

Parameter Name	**Parameter Value**
Packet header length	4 bytes
Network Layer packet header format and associated Reconfigurable Components addresses	Defined by the RRRM at runtime

Figure 5.9 contains a description of the operations performed by the Header Routing Reconfigurable PMUX Algorithm which can be combined with the internal operations description provided in Chapter 3 to produce a complete representation of how the PMUX was implemented in the CRRSA implementation.

5.3.2.3 Implementing a Reconfigurable Waveform Multiplexing Algorithm

The Reconfigurable Waveform Multiplexing (WMUX) Algorithm is responsible for multiplexing all RATs' waveforms onto a single waveform which can be transmitted by the SDR hardware platform, and Demultiplexing received waveforms to retrieve each RAT's original waveform. The Reconfigurable WMUX Algorithm developed for the CRRSA implementation utilises Frequency Division Multiplexing (FDM) to perform its tasks and is thus called the FDM Reconfigurable WMUX Algorithm.

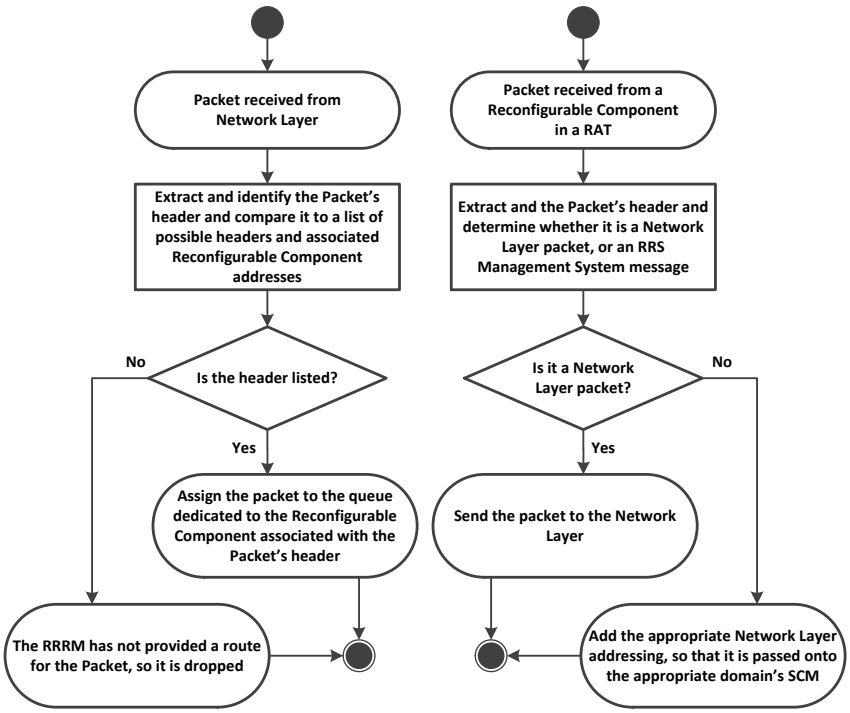


Figure 5.9: The operations performed by the Header Routing Reconfigurable Packet Multiplexing Algorithm

As depicted in Figure 5.10, the FDM Reconfigurable WMUX Algorithm modulates each RAT's waveform onto a separate frequency band, thereby forming a single waveform which is transmitted by the SDR. This process is reversed at the receiver, where each RAT's waveform is demodulated from the SDR waveform. (See [16,126])

The RRRM determines the carrier frequencies for each RAT waveform based on the waveform's bandwidth specified in its RAT policy, and then instructs the FDM Reconfigurable WMUX Algorithm to use the calculated carrier frequencies when modulating waveforms sent from specific Reconfigurable Components in each RAT. Although the FDM Algorithm only provides the RRRM with limited control over spectrum allocation, the functionality provided is sufficient to demonstrate that the RRRM is able to exercise control over the waveform transmission process, and more advanced Reconfigurable WMUX Algorithms could be implemented to support more complex DSA applications.

Before the FDM operation is performed each RAT waveform is upsampled and pulse shaped using a Root Raised Cosine Filter. This process is necessary as it

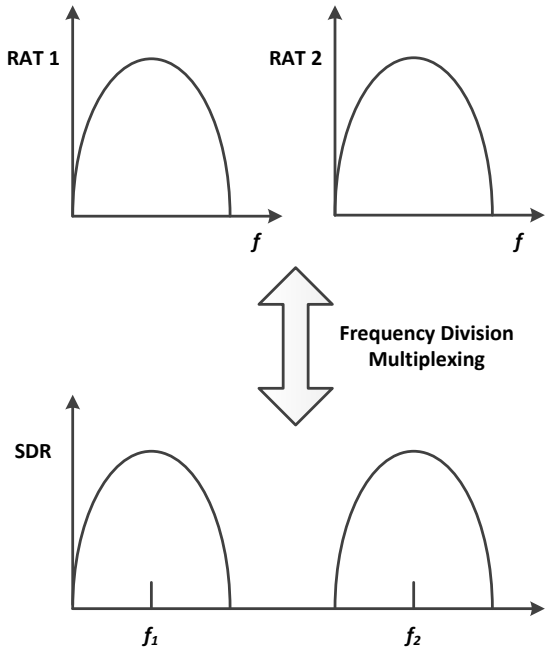


Figure 5.10: Frequency Division Multiplexing (Derived from [16, 126])

limits the bandwidth of each waveform thereby preventing intersymbol interference. The process is reversed at the receiver using a matched Root Raised Cosine Filter. (See [16, 126])

The final step in the FDM process requires the receiver to identify whether a signal was actually transmitted in a given frequency band. This operation is required as different RATs may produce waveforms for transmission at different rates. To prevent faster RATs from having to wait for slower RATs to produce waveforms, the WMUX multiplexes all available RAT waveforms whenever the SDR API indicates that it is ready to transmit a signal. The FDM Reconfigurable WMUX Algorithm determines whether a signal was transmitted by calculating the average power of each demodulated RAT waveform, and compares it with a predefined signal detection threshold. If a waveform is transmitted, the received RAT waveform's average power will exceed the threshold, and the waveform is then sent to the relevant RAT.

The input parameters and values used in the CRRSA implementation for the FDM Reconfigurable WMUX Algorithm's FDM operations are presented in Table 5.3. These values were calculated using the NyquistShannon sampling theorem as well as the bandwidth and power of the signals produced by the RATs in the CRRSA implementation.

Table 5.3: The FDM Reconfigurable Waveform Multiplexing Algorithm's FDM input parameters for the CRRSA implementation

Parameter Name	Parameter Value
Sampling Frequency	5 MHz
Pulse Shaping Filter Order	6
Pulse Shaping Filter Roll-off Factor	0.5
Pulse Shaping Filter Upsampling Factor	32
Carrier Frequency for each RAT's Waveform and the addresses of the Reconfigurable Components that interface with the WMUX	Defined by RRRM at runtime
Signal Detection Threshold	0.0001

In addition to performing FDM, the FDM Reconfigurable WMUX Algorithm is also responsible for synchronising each received signal, by identifying the start of each physical layer frame within the time domain, and by correcting any frequency offsets resulting from transmission through the wireless environment. The FDM Algorithm performs this function using the maximum likelihood time and frequency offset estimator published by van de Beek et al. in 1997 [127]. This estimator appends an OFDM symbol to each transmitted waveform, which is used at the receiver to identify the start of the physical layer frame and estimate the frequency offset. Table 5.4 presents the synchronisation related input parameters and values used in the CRRSA implementation.

Table 5.4: The FDM Reconfigurable Waveform Multiplexing Algorithm's synchronisation input parameters for the CRRSA implementation

Parameter Name	Parameter Value
Number of Subcarriers	32
Subcarrier Spacing	156.25 kHz
Cyclic Prefix Ratio	0.25
Number of Samples per Transmitted Signal	93296

The FDM Reconfigurable WMUX Algorithm's FDM and synchronisation calculations are implemented in MATLAB functions. These functions interface with the FDM Algorithm using the MATLAB Control API, as the Root Raised Cosine filter functions provided by the MATLAB Signal Processing Toolbox cannot be compiled using the MATLAB Builder JA. Figure 5.11 provides an overview of

the operations performed by the FDM Reconfigurable WMUX Algorithm and its MATLAB functions.

5.3.2.4 Implementing a Reconfigurable Decision Making Algorithm

The Reconfigurable Decision Making Algorithm determines how reconfiguration decisions are made in the CRRSA, using context information supplied by the RMM regarding the status and performance of existing RATs, and performance metric specifications for available RATs defined in the policies. The Reconfigurable Decision Making Algorithm developed for the CRRSA implementation is designed to showcase the functionality provided by the CRRSA, by performing reconfiguration operations according to a set of predefined steps, and is referred to as the Basic Management Demo Reconfigurable Decision Making Algorithm.

To enable the Basic Management Demo Reconfigurable Decision Making Algorithm to interface with the Header Routing and FDM Algorithms in the PMUX and WMUX, it was developed in conjunction with these Algorithms and provided with the information needed to calculate the input parameters for each of the Algorithms. This process is necessary when developing any Reconfigurable Decision Making Algorithm and it does not detract from the overall flexibility offered by the CRRSA. The Reconfigurable Algorithms implemented in any CRSSA implementation can be thought of as a combined set, which jointly provide a set of capabilities which exploit and expand the basic reconfiguration management functionality provided by the CRRSA.

The Basic Management Demo Reconfigurable Decision Making Algorithm implements the reconfiguration processes exactly as defined in Chapter 3. It initialises all of the CRRSA's reconfigurable management system's logical components at the start of the simulation and instructs the RC to create the default RAT. After this initial process has been completed, the Basic Management Demo Reconfigurable Decision Making Algorithm in each domain continues to monitor the status of their default RAT using context information obtained from the RMM and performs the following functions:

1. When the Basic Management Demo Reconfigurable Decision Making Algorithm in the Subscriber domain detects that the default RAT has received a predefined number of packets, it submits a request to the Access Network domain's Basic Management Demo Reconfigurable Decision Making Algorithm

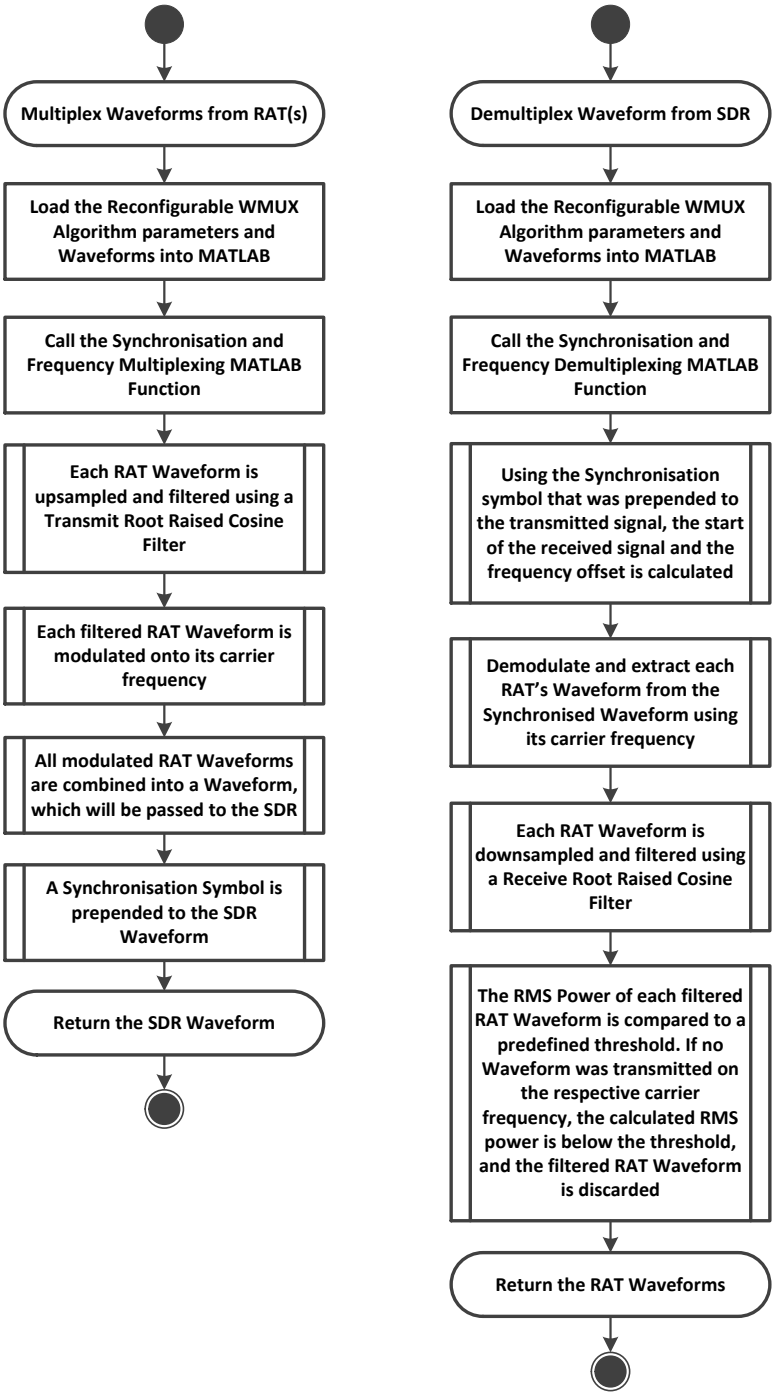


Figure 5.11: The operations performed by the FDM Reconfigurable Waveform Multiplexing Algorithm

to create a second RAT. A predefined number of 20 packets were chosen for the implementation, in order to ensure that the entire simulation was able to be run within a realistic time period.

2. The Subscriber domain's request is accepted by the Access Network domain's Basic Management Demo Reconfigurable Decision Making Algorithm, and the Algorithms in both domains then instruct their respective RCs to create the new RAT and adjust the parameters in their PMUXs and WMUXs accordingly.

3. After monitoring the second RAT's status, and determining that it has also received a predefined number of packets, the Access Network domain's Basic Management Demo Reconfigurable Decision Making Algorithm submits a request to the Subscriber domain's Basic Management Demo Reconfigurable Decision Making Algorithm to terminate the first RAT.

4. This request is accepted, and both Basic Management Demo Reconfigurable Decision Making Algorithms instruct their RCs to terminate the original RAT and again adjust the parameters in their PMUXs and WMUXs accordingly.

The reason for developing the Basic Management Demo Reconfigurable Decision Making Algorithm in this manner is to provide a deterministic series of events to demonstrate each of the primary reconfiguration management operations provided by CRRSA. The policies used by the Basic Management Demo Reconfigurable Decision Making Algorithm to carry out its operations are defined in the following section, and the results of this process and the management operations carried out by the entire CRRSA implementation are provided in Section 5.4.

5.3.3 Defining Network Equipment and RAT Policies

The CRRSA implementation defines separate Network Equipment Policies for a Mobile Device in the Subscriber domain, and a Base Station in the Access Network domain. These Network Equipment Policies are implemented in XML and provide the RRRM with a list of available RATs and the names and parameter values for each of the Reconfigurable Algorithm implementations previously discussed. The list of RATs which are available for implementation include the 802.16 and LTE RAT models developed in Chapter 4. The 802.16 RAT is defined to be the default RAT which must be implemented immediately after each of the Subscriber and Access Network domains have booted up.

Like the Network Equipment policies, the 802.16 and LTE RAT models are also defined in separate RAT policies for the Subscriber and Access Network domains. Although the CRRSA implementation utilises the same RAT models for each domain, the formats for the different types of policies are different to reflect the fact that they are designed to be used in separate domains.

Each RAT policy specifies the performance metrics for its RAT, the names of the Reconfigurable Components needed to construct the RAT, and the connections between the input and output interfaces for all Reconfigurable Components. Table 5.5 presents the performance metrics contained in each of the RAT policies. These values were calculated based on the digital modulation schemes, FEC coding rates, the number of OFDM subcarriers, the spacing between subcarriers, and the sampling rate used in the implementation of each of the RATs.

Table 5.5: Performance Metrics contained in the CRRSA implementation's RAT Policies

Performance Metrics	**802.16 (WiMAX)**	**LTE**
Transmission Bandwidth	2.048 MHz	2.048 MHz
Carrier Frequency	5 GHz	5 GHz
Max. Transmission Power	40 dB	40 dB
Max. Uplink Throughput	79.02 kbps	105.37 kbps
Max. Downlink Throughput		
Uplink Spectral Efficiency	0.0386 bits/s/Hz	0.0514 bits/s/Hz
Downlink Spectral Efficiency		
Duplexing Scheme	TDD	TDD

Each of the Network Equipment and RAT policies have their own unique ID number, which enables the RPMs in the Subscriber and Access Network domains to determine whether such a policy already exists in their policy database, or whether a policy needs to be downloaded from the Core Network domain's policy database. Table 5.6 presents the ID numbers for the policies used in the CRRSA implementation.

Table 5.6: The ID Numbers for the Network Equipment and RAT Policies used in the CRRSA implementation

Type of Policy	**ID Number in the Access Network domain**	**ID Number in the Subscriber domain**
Network Equipment Policy	NE00001	NE00002
LTE RAT Policy	RAT00001	RAT00002
802.16 RAT Policy	RAT00003	RAT00004

5.3.4 Implementing Reconfigurable Components

The CRRSA implementation constructs RATs using Reconfigurable Components implemented in JADE agents. When each agent is launched by the RC during the RAT creation process, the RC provides the addresses of the other Reconfigurable Component agents which provides the agent with its required inputs. After this operation has been completed, the agent must create instances of two Behaviour classes which manage the acquisition and processing of inputs and also control the distribution of outputs. Each of these Behaviour instances runs in a separate thread, in order to enable them to perform their operations simultaneously.

The specific context information submitted to the RMM by Reconfigurable Components in the CRRSA implementation is intended to demonstrate how the performance of a RAT can be monitored and how this performance data can used by the RRRM to make reconfiguration decisions. Depending on the nature of the functionality implemented by the Reconfigurable Component, different RAT performance measurements can be collected and the results submitted to the RMM. It is important to note that the CRRSA does not place any limits on what measurements can be taken and submitted to the RMM; however all measurements must be submitted in a format which can be read and stored by the RMM. The CRRSA implementation defines the following five different types of RAT performance measurements, and provides methods for Reconfigurable Components to submit values to the RMM:

- The number of packets received by a Reconfigurable Component.
- The number of bits received by a Reconfigurable Component.
- The number of packets received by a Reconfigurable Component which have been identified as corrupted.
- The average processing time take to execute a Reconfigurable Component's atomic function.
- The Received Signal Strength Indicator (RSSI), which is measured by the SDR hardware platform and supplied to a requesting Reconfigurable Component by the WMUX.

When the RC instructs a Reconfigurable Component to terminate itself, the Reconfigurable Component's agent waits until both of the Behaviours have completed their current tasks before stopping both threads. Once the Behaviour threads have

been stopped, the agent instructs the JADE platform to deregister its address and to stop its thread.

The generic and RAT specific functions developed in Chapter 4 were encapsulated in Reconfigurable Component agents and implemented in Java and MATLAB functions. Appendix B contains a list of all the generic and RAT specific Reconfigurable Components used to construct RATs in the CRRSA implementation, and specifies each Component's input and output parameters, implementation language and/or method, and the context information submitted to the RMM.

The Abstracted Control Reconfigurable Components also defined in Chapter 4, provide statically defined and random parameters to the generic and RAT specific Reconfigurable Components in each RAT. The specific parameters provided by the Abstracted Control Reconfigurable Components for the 802.16 and LTE RATs are also listed in Appendix B.

5.3.5 Implementing the SDR API

The SDR API developed for the CRRSA implementation is a JADE agent which interfaces with the WMUXs in both the Subscriber and Access Network domains using the interface definition provided in Chapter 3. The SDR API agent supports all three Wireless Channel Scenarios via the same interface, by implementing each Scenario's functionality in a separate Behaviour class. During the simulation initialisation process, the user chooses which Wireless Channel Scenario should be used for the duration of the simulation, and provides the transmission parameters for the chosen Scenario. The purpose of this functionality is to allow the CRRSA's reconfigurable management system and RAT implementations to be tested under a variety of different simulated and real channel conditions.

Figure 5.12 presents the architecture of the SDR API and illustrates how each Wireless Channel Scenario is incorporated into the implementation. The SDR API JADE agent logically separates the uplink and downlink channels by storing waveforms in separate buffers. Depending on which Wireless Channel Scenario is being used, the waveforms stored in the transmit buffers are either transferred directly to their respective receive buffers and passed through a simulated Wireless Channel using the MATLAB Channel Simulation Function and the MATLAB Control API, or these waveforms are transmitted over the wireless environment using two USRP2 SDR hardware platforms via a C++ SDR API Wrapper class, the Java Native Interface,

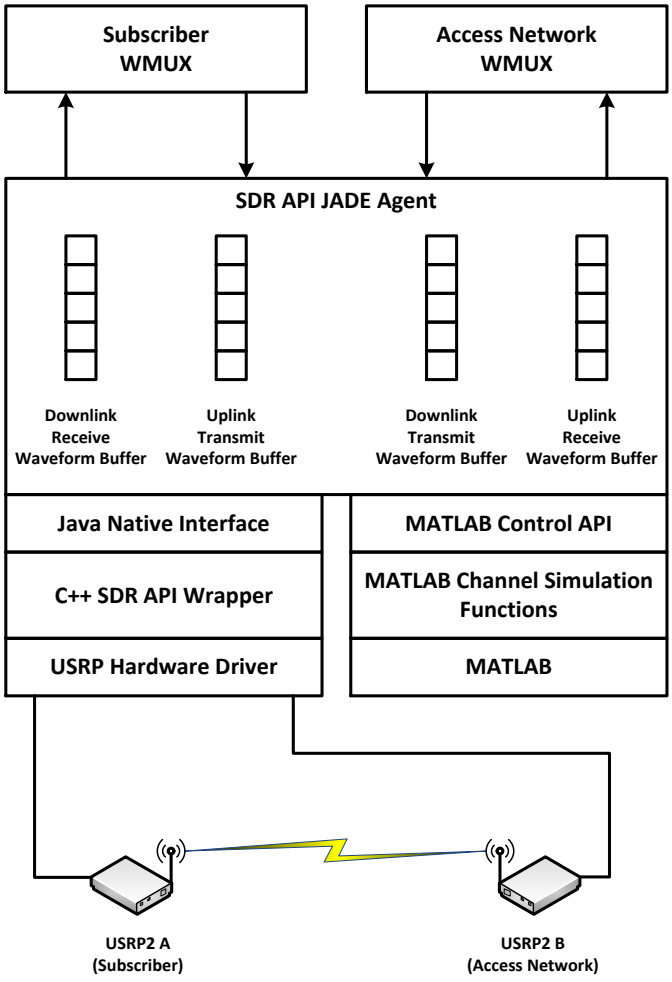


Figure 5.12: The SDR API Architecture for the CRRSA implementation

and the UHD. Figure 5.13 provides a detailed description of the operations performed by the SDR API agent for each of the three Wireless Channel Scenarios.

Although the user can change the Wireless Channel Scenarios' parameters during the simulation's initialisation phase, a set of default parameters is provided for convenience. Table 5.7 presents the default transmission parameters for the MATLAB Simulated Wireless Channel Scenario. These parameters are designed to demonstrate that the implemented RATs' can successfully transmit a signal carrying data in an indoor environment and within a frequency band supported by both the 802.16 and LTE standards.

The UHD provides a set of functions to initialise the transmission parameters for a USRP, to transmit an RF signal, and to monitor a particular frequency band and

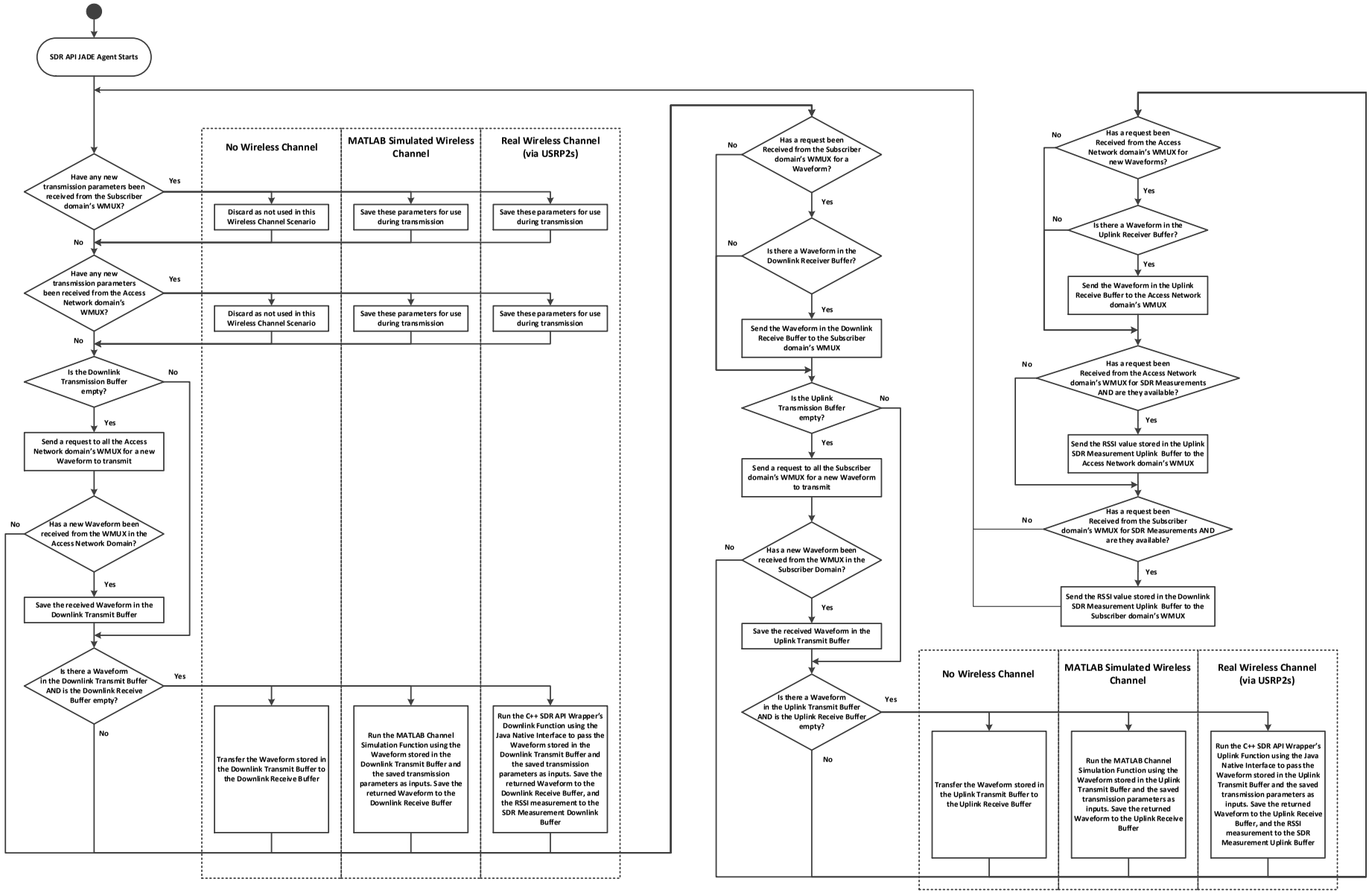


Figure 5.13: The operations performed by the SDR API JADE agent in each of the three Wireless Channel Scenarios

Table 5.7: Default Transmission Parameters for the MATLAB Simulated Wireless Channel Scenario

MATLAB Wireless Simulation Channel Parameter	**Parameter Values (Subscriber and Access Network domains)**
Signal to Noise Ratio	20 dB
Transmit Signal Power	10 mW
Sampling Rate	5 MHz
Carrier Frequency	5 GHz
Terminal Speed	0 m/s
Fading Delays	[0 4 8 12] ns
Path Gains	[0 -3 -6 -9] dB

save the resulting RF signal. The SDR API C++ Wrapper provides a framework for initialising, managing and coordinating the operations of the two USRP2s using the functions provided by the UHD. The SDR API C++ Wrapper is compiled as a 64-bit library in the Windows 7 operating system.

Prior to each transmission, the SDR API JADE agent specifies the transmission parameters to be used, thereby providing a granular degree of control over the transmission process. During transmission operations, the SDR API C++ Wrapper implements two threads to control the transmitting and receiving USRP2s separately. As the XCVR2450 daughterboards used in the CRRSA implementation only support TDD, the SDR API C++ Wrapper only provides the functionality to perform uplink or downlink transmissions separately. The operations performed by the SDR API C++ Wrapper are described in more detail in Figure 5.14.

As is the case with the MATLAB Simulated Wireless Channel Scenario, the Real Wireless Channel Scenario also possesses a set of default transmission parameters. These parameters are designed to match those of the MATLAB Simulated Wireless Channel Scenario as closely as possible, and are presented in Table 5.8. The Real Wireless Channel Scenario simulation was carried out in an indoor environment, and the results from this simulation are presented along with those of the MATLAB Simulated Wireless Channel Scenario simulation in the following section.

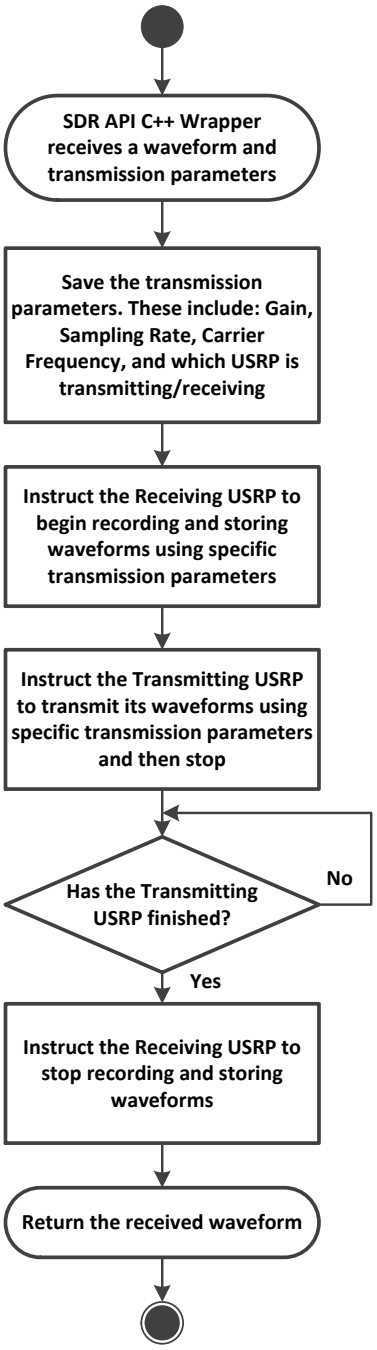


Figure 5.14: The operations performed by the C++ SDR API Wrapper

Table 5.8: Default Transmission Parameters for the Real Wireless Channel Scenario

USRP2 Transmission Parameters	**Parameter Values for USRP2 A**	**Parameter Values for USRP2 B**
IP Addresses	192.168.10.2	192.168.20.2
Transmit Gain	30 dB	30 dB
Receive Gain	40 dB	40 dB
Transmit Sampling Rate	5 MHz	5 MHz
Receive Sampling Rate	5 MHz	5 MHz
Transmit Carrier Frequency	5 GHz	5 GHz
Receive Carrier Frequency	5 GHz	5 GHz

5.4 Results and Critical Analysis

The following subsections present the results from a series of simulations which were performed to demonstrate the functionality provided by the CRRSA, and in doing so prove that it fulfils the implementation objectives defined in Section 5.1

5.4.1 Management Functionality Results

To demonstrate the functionality provided by the CRRSA's reconfigurable management system and operating environment, a No Wireless Channel Scenario simulation was performed using the implemented Network Equipment and RAT Policies, the Reconfigurable Components, and Reconfigurable Algorithms described in the previous sections of this chapter.

This section provides a detailed description of each phases of this simulation, and describes the operations carried out by the CRRSA's logical components and the messages passed between them in order to first initialise the reconfigurable management system in each domain, and then to make and execute reconfiguration decisions using the supplied policies and context information from the implemented RATs.

After the simulation parameters were entered, and the Core Network domain's agents were launched, the Core Network domain's reconfigurable management system executed the operations illustrated in Figure 5.15, which are described as follows:

1. The RCAL compiles and validates all the Reconfigurable Components and Algorithms implemented in Java files and stored in the folder representing the

Core Network domain's Reconfigurable Component and Algorithms database.

2. The RCAL sends the RPM a list of all available Reconfigurable Components and Algorithms.

3. The RPM uses this list to validate the Network Equipment and RAT policies implemented in XML files and stored in the folder representing the Core Network domain's Reconfigurable Policy database.

4. The RPM examines the available Network Equipment policies and determines that the SCM should use the Stop and Wait ARQ Reconfigurable Communications Algorithm. This information is then passed onto the RCAL.

5. The RCAL then provides the SCM with an instance of the Stop and Wait ARQ Reconfigurable Communications Algorithm.

6. The SCM loads the Stop and Wait ARQ Reconfigurable Communications Algorithm using default parameters.

7. The Core Network domain's reconfigurable management system sits idle until contacted by one of the other domains.

After the Access Network domain's reconfigurable management system has launched, the Access Network domain's RPM and RCAL determine that there are no policies, Reconfigurable Components, or Reconfigurable Algorithms available, and then they submit requests to the Core Network domain which are processed as illustrated in Figure 5.15 and described as follows:

1. The Core Network domain's SCM receives a series of packets using its Stop and Wait ARQ Reconfigurable Communications Algorithm. These packets contain the Access Network domain's RPM's request for new Network Equipment and RAT policies.

2. The SCM reassembles this request and relays it to the Core Network domain's RPM.

3. The RPM examines the list of policies that the Access Network domain already possesses and determines that the Access Network does not initially have any policies.

4. The RPM then selects the NE00001, RAT00001, and RAT00003 Network Equipment and RAT policies, encapsulates them in a reconfigurable management message, and sends it to the Core Network domain's SCM.

5. The Core Network domain's SCM splits the message into a series of packets and transmits them to the Access Network domain's SCM using its Stop and Wait ARQ Reconfigurable Communications Algorithm.

6. The Core Network domain's SCM then receives a new series of packets using its Stop and Wait ARQ Reconfigurable Communications Algorithm. These new packets contain the Access Network domain's RCAL's request for new Reconfigurable Components and/or Algorithms.

7. The SCM reassembles this request and relays it to the Core Network domain's RCAL.

8. The RCAL examines the list of Reconfigurable Components and/or Algorithms that the Access Network domain already possesses and determines that the Access Network does not have any Reconfigurable Components or Algorithms.

9. The RCAL then selects all the Reconfigurable Components and Algorithms described in this Chapter, encapsulates them in a reconfigurable management message, and sends it to the Core Network domain's SCM.

10. The Core Network domain's SCM splits the message into a series of packets and transmits them to the Access Network domain's SCM using its Stop and Wait ARQ Reconfigurable Communications Algorithm.

After the Subscriber domain's reconfigurable management system has launched, the default RAT has been implemented, and a wireless connection has been established with the Access Network domain, the Subscriber domain's RPM and RCAL submit requests to the Core Network domain which are processed as illustrated in Figure 5.15 and described as follows:

1. The Core Network domain's SCM receives a series of packets using its Stop and Wait ARQ Reconfigurable Communications Algorithm. These packets contain the Subscriber domain's RPM's request for new Network Equipment and RAT policies.

2. The SCM reassembles this request and relays it to the Core Network domain's RPM.

3. The RPM examines the list of policies that the Subscriber domain already possesses and determines that there are no new policies to send to the Subscriber domain's RPM, which results in the Core Network domain's RPM ignoring the request.

4. The Core Network domain's SCM then receives a new series of packets using its Stop and Wait ARQ Reconfigurable Communications Algorithm. These new packets contain the Subscriber domain's RCAL's request for new Reconfigurable Components and/or Algorithms.

5. The SCM reassembles this request and relays it to the Core Network domain's RCAL.

6. The RCAL examines the list of Reconfigurable Components and/or Algorithms that the Access Network domain already possesses and determines that there are no new Reconfigurable Components or Algorithms to send to the Subscriber domain's RCAL, which results in the Core Network domain's RCAL ignoring the request.

After the Core Network domain's reconfigurable management system has completed its initial operations, the Access Network domain's reconfigurable management system is launched and executes the operations illustrated in Figure 5.16, which are described as follows:

1. The Access Network domain's SCM loads the default Stop and Wait ARQ Reconfigurable Communications Algorithm and then waits for any communications operations to occur.

2. The RCAL examines the folder representing the Access Network domain's Reconfigurable Component and Algorithms database, and determines that there are no Reconfigurable Components or Algorithms available.

3. The RCAL submits a request for new Reconfigurable Components and/or Algorithms from the Core Network to the Access Network's SCM.

4. The Access Network domain's SCM splits the message into a series of packets and transmits them to the Core Network domain's SCM using its Stop and Wait ARQ Reconfigurable Communications Algorithm.

5. The Core Network domain's SCM receives the packets, reassembles the message, and forwards it to the Core Network domain's RCAL. The Core Network domain's RCAL then encapsulates the new Reconfigurable Components and Algorithms in a message and sends it to the Core Network domain's SCM. The Core Network domain's SCM splits the message into packets and sends them to the Access Network domain's SCM.

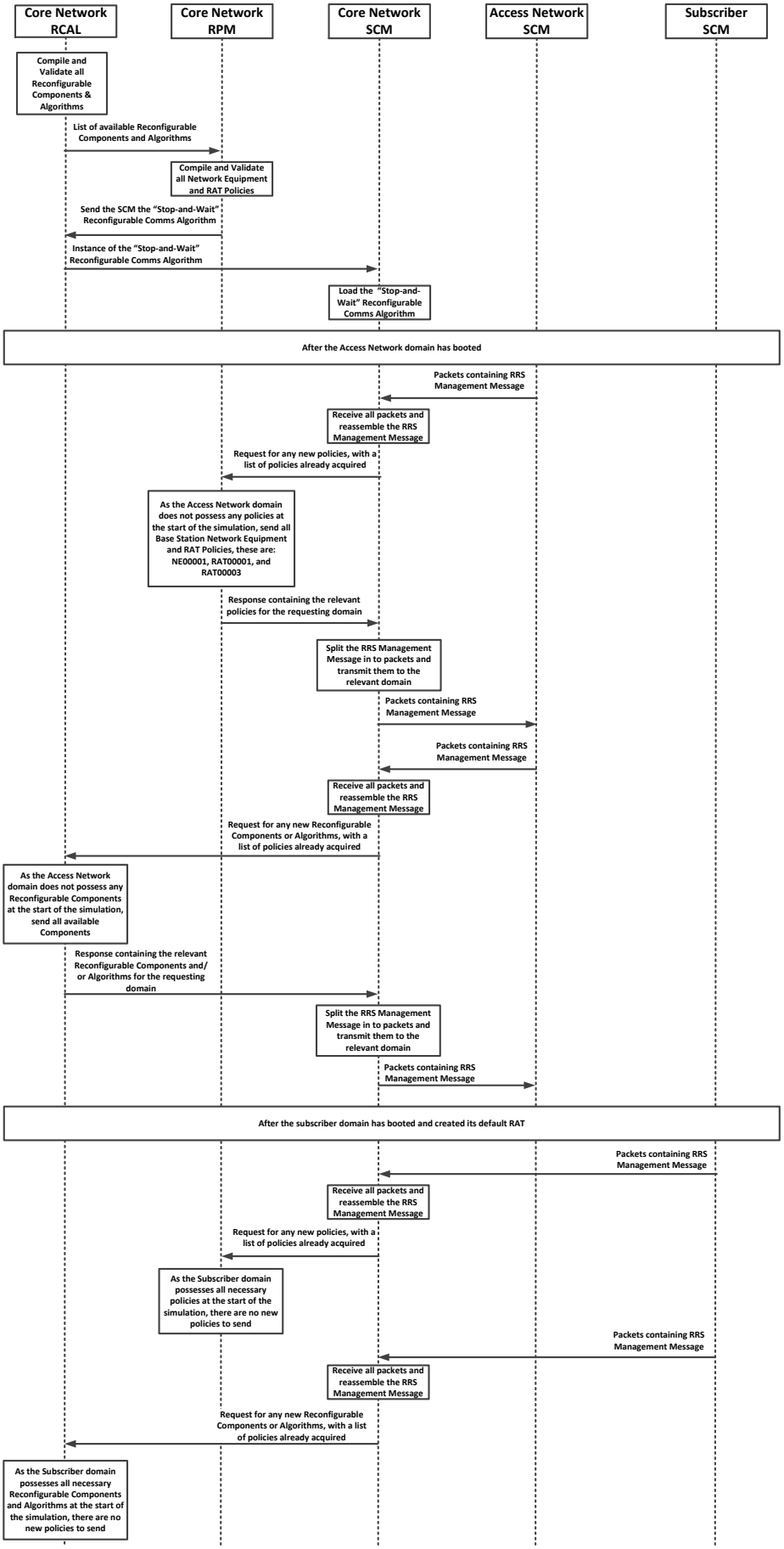


Figure 5.15: Reconfigurable Management System operations performed in the Core Network domain during the CRRSA implementation simulation

6. The Access Network domain's SCM reassembles the message and passes it to the Access Network domain's RCAL.

7. The RCAL stores the received Reconfigurable Components and Algorithms in the folder representing the Access Network domain's Reconfigurable Component and Algorithms database, and compiles and validates all of the new files.

8. The RCAL sends the RPM a list of all available Reconfigurable Components and Algorithms.

9. The RPM examines the folder representing the Access Network domain's Reconfigurable Policy database, and determines that there are no policies available.

10. The RPM submits a request for new policies from the Core Network to the Access Network's SCM.

11. The Access Network domain's SCM splits the message into a series of packets and transmits them to the Core Network domain's SCM using its Stop and Wait ARQ Reconfigurable Communications Algorithm.

12. The Core Network domain's SCM receives the packets, reassembles the message, and forwards it to the Core Network domain's RPM. The Core Network domain's RPM then encapsulates the new policies in a message and sends it to the Core Network domain's SCM. The Core Network domain's SCM splits the message into packets and sends them to the Access Network domain's SCM.

13. The Access Network domain's SCM reassembles the message and passes it to the Access Network domain's RPM.

14. The RPM stores the received Network Equipment and RAT Policies in the folder representing the Access Network domain's Reconfigurable Component and Algorithms database, and compiles and validates all of the new files using the list of available Reconfigurable Components and Algorithms previously provided by the RCAL.

15. The RPM sends to NE00001 Network Equipment policy to the RRRM for implementation.

16. The RRRM examines the NE00001 Network Equipment policy and determines that the Basic Management Demo Reconfigurable Decision Making Algorithm should be implemented.

17. The RRRM submits a request to the RCAL for an instance of the Basic Management Demo Reconfigurable Decision Making Algorithm.

18. The RCAL responds with an instance of the Basic Management Demo Reconfigurable Decision Making Algorithm.

19. The RRRM loads the Basic Management Demo Reconfigurable Decision Making Algorithm and uses it to begin the core initialisation procedures according to the specifications in the NE00001 Network Equipment Policy, which specifies that new Reconfigurable Algorithms are to be loaded into the PMUX and WMUX. The SCM's default Reconfigurable Communications Algorithm is specified by the NE00001 Network Equipment Policy, however as it was already implemented, no further action is taken by the RRRM.

20. The RRRM sends a request to the RCAL for an instance of the Header Routing Reconfigurable PMUX Algorithm, which is defined in the NE00001 Network Equipment Policy as the default Reconfigurable Algorithm for the PMUX.

21. The RCAL responds with an instance of the requested Header Routing Reconfigurable PMUX Algorithm.

22. The RRRM sends the instance of the Header Routing Reconfigurable PMUX Algorithm and the default parameters defined by the NE00001 Network Equipment Policy to the PMUX.

23. The PMUX receives and loads the instance of the Header Routing Reconfigurable PMUX Algorithm with the supplied default parameters.

24. The RRRM sends a request to the RCAL for an instance of the FDM Reconfigurable WMUX Algorithm, which is defined in the NE00001 Network Equipment Policy as the default Reconfigurable Algorithm for the WMUX.

25. The RCAL responds with an instance of the requested FDM Reconfigurable WMUX Algorithm.

26. The RRRM sends the instance of the FDM Reconfigurable WMUX Algorithm and the default parameters defined by the NE00001 Network Equipment Policy to the WMUX.

27. The WMUX receives and loads the instance of FDM Reconfigurable WMUX Algorithm with the supplied default parameters.

28. The RRRM instructs the RC to create the default RAT listed by the NE00001 Network Equipment Policy, using the RAT00003 RAT Policy which contains the specification for the 802.16 RAT model.

29. The RC receives the instruction and sends a request to the RPM for a copy of the RAT00003 RAT Policy

30. The RPM sends the RC a copy of the RAT00003 RAT Policy.

31. The RC examines the RAT00003 RAT Policy, and requests instances of the specified Reconfigurable Components from the RCAL.

32. The RCAL responds with the requested Reconfigurable Component instances.

33. The RC encapsulates the Reconfigurable Component instances in agents, assigns each agent a unique address, and provides the connection information needed by each agent to acquire inputs and distribute outputs.

34. The RC informs the RRRM that the RAT has been created, and provides a list of all the Reconfigurable Component agent's addresses.

35. The RRRM stores the addresses, and also sends them to the RMM.

36. The RMM stores the received addresses, and uses the information to determine which Reconfigurable Components belong to each RAT. This is necessary as the RMM constantly receives context information from Reconfigurable Components, and each item of information must be attributed to a particular RAT so that when the RMM compiles its context information report for the RRRM, it knows which items of information to use for each RAT.

37. The RRRM calculates the new parameters for the Header Routing Reconfigurable PMUX Algorithm, which determines how traffic is scheduled between the existing RATs, and then sends them to the PMUX. At this point in the simulation, the RRRM instructs the PMUX to route all traffic through the default RAT as there is no other existing RAT.

38. The PMUX receives the new parameters and stores them for use by the Header Routing Reconfigurable PMUX Algorithm.

39. The RRRM calculates the new parameters for the FDM Reconfigurable WMUX Algorithm, which determines the carrier frequency for each RAT, and then sends this information to the WMUX.

40. The WMUX receives the new parameters and stores them for use by the FDM Reconfigurable WMUX Algorithm.

After the Access Network domain's reconfigurable management system has completed its initial operations, the Subscriber domain's reconfigurable management

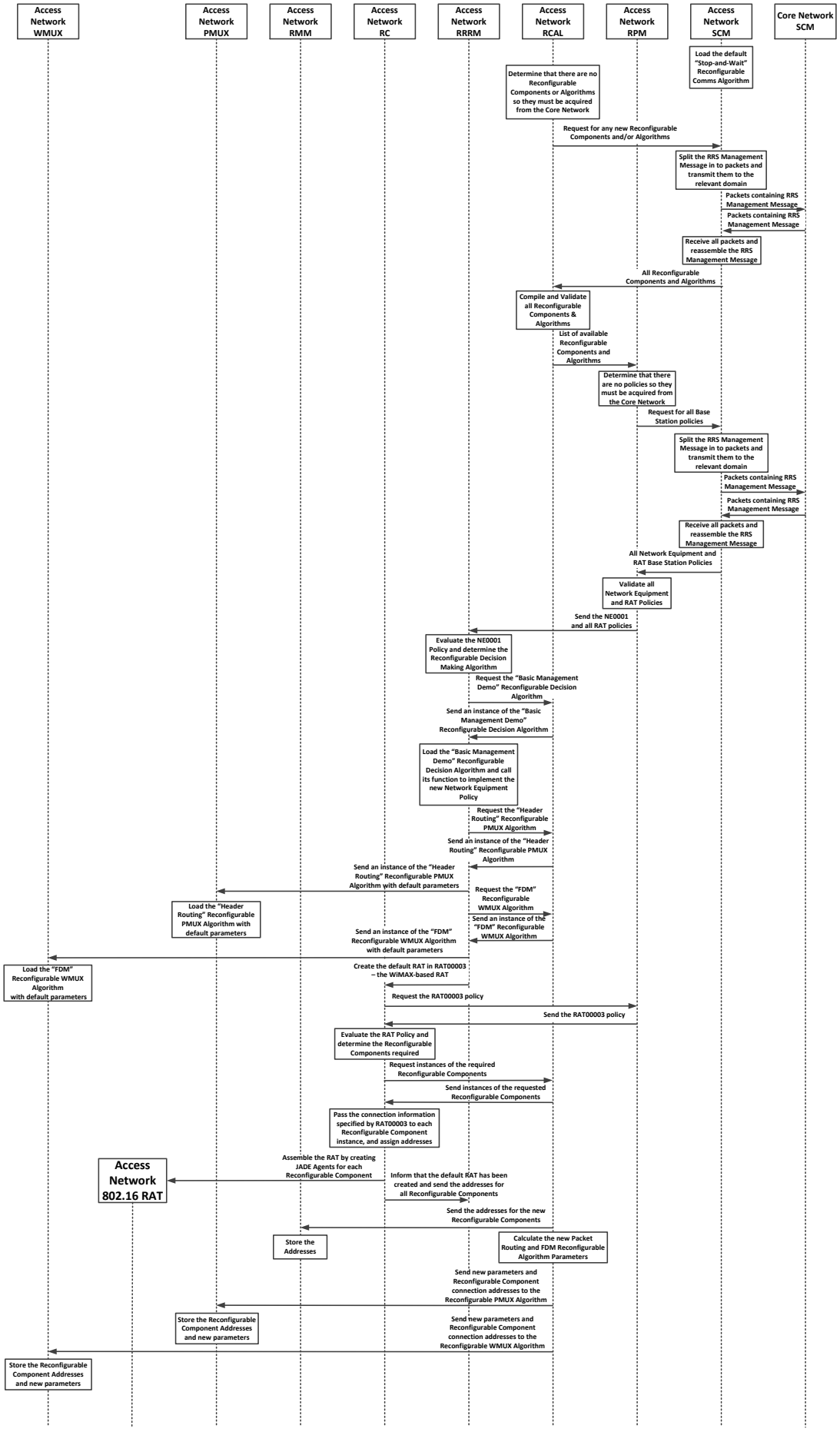


Figure 5.16: Reconfigurable Management System operations performed in the Access Network domain during the initial phases of the CRRSA implementation simulation

system is launched and executes the operations illustrated in Figure 5.17, which are described as follows:

1. The Subscriber domain's SCM loads the default Stop and Wait ARQ Reconfigurable Communications Algorithm and then waits for any communications operations to occur.

2. The RCAL compiles and validates all the Reconfigurable Components and Algorithms implemented in Java files and stored in the folder representing the Subscriber domain's Reconfigurable Component and Algorithms database.

3. The RCAL sends the RPM a list of all available Reconfigurable Components and Algorithms.

4. The RPM uses this list to validate the Network Equipment and RAT policies implemented in XML files and stored in the folder representing the Subscriber domain's Reconfigurable Policy database.

5. The RPM sends to NE00002 Network Equipment policy to the RRRM for implementation.

6. The RRRM examines the NE00002 Network Equipment policy and determines that the Basic Management Demo Reconfigurable Decision Making Algorithm should be implemented.

7. The RRRM submits a request to the RCAL for an instance of the Basic Management Demo Reconfigurable Decision Making Algorithm.

8. The RCAL responds with an instance of the Basic Management Demo Reconfigurable Decision Making Algorithm.

9. The RRRM loads the Basic Management Demo Reconfigurable Decision Making Algorithm and uses it to begin the core initialisation procedures according to the specifications in the NE00002 Network Equipment Policy, which specifies that new Reconfigurable Algorithms are to be loaded into the PMUX and WMUX. The SCM's default Reconfigurable Communications Algorithm is specified by the NE00002 Network Equipment Policy, however as it was already implemented, no further action is taken by the RRRM.

10. The RRRM sends a request to the RCAL for an instance of the Header Routing Reconfigurable PMUX Algorithm, which is defined in the NE00002 Network Equipment Policy as the default Reconfigurable Algorithm for the PMUX.

11. The RCAL responds with an instance of the requested Header Routing Reconfigurable PMUX Algorithm.

12. The RRRM sends the instance of the Header Routing Reconfigurable PMUX Algorithm and the default parameters defined by the NE00002 Network Equipment Policy to the PMUX.

13. The PMUX receives and loads the instance of the Header Routing Reconfigurable PMUX Algorithm with the supplied default parameters.

14. The RRRM sends a request to the RCAL for an instance of the FDM Reconfigurable WMUX Algorithm, which is defined in the NE00002 Network Equipment Policy as the default Reconfigurable Algorithm for the WMUX.

15. The RCAL responds with an instance of the requested FDM Reconfigurable WMUX Algorithm.

16. The RRRM sends the instance of the FDM Reconfigurable WMUX Algorithm and the default parameters defined by the NE00002 Network Equipment Policy to the WMUX.

17. The WMUX receives and loads the instance of FDM Reconfigurable WMUX Algorithm with the supplied default parameters.

18. The RRRM instructs the RC to create the default RAT listed by the NE00001 Network Equipment Policy, using the RAT00004 RAT Policy which contains the specification for the 802.16 RAT model.

19. The RC receives the instruction and sends a request to the RPM for a copy of the RAT00004 RAT Policy

20. The RPM sends the RC a copy of the RAT00004 RAT Policy.

21. The RC examines the RAT00004 RAT Policy, and requests instances of the specified Reconfigurable Components from the RCAL.

22. The RCAL responds with the requested Reconfigurable Component instances.

23. The RC encapsulates the Reconfigurable Component instances in agents, assigns each agent a unique address, and provides the connection information needed by each agent to acquire inputs and distribute outputs.

24. The RC informs the RRRM that the RAT has been created, and provides a list of all the Reconfigurable Component agent's addresses.

25. The RRRM stores the addresses, and also sends them to the RMM.

26. The RMM stores the received addresses, and uses the information to determine which Reconfigurable Components belong to each RAT. This is necessary as the RMM constantly receives context information from Reconfigurable Components, and each item of information must be attributed to a particular RAT so that when the RMM compiles its context information report for the RRRM, it knows which items of information to use for each RAT.

27. The RRRM calculates the new parameters for the Header Routing Reconfigurable PMUX Algorithm, which determines how traffic is scheduled between the existing RATs, and then sends them to the PMUX. At this point in the simulation, the RRRM instructs the PMUX to route all traffic through the default RAT as there is no other existing RAT.

28. The PMUX receives the new parameters and stores them for use by the Header Routing Reconfigurable PMUX Algorithm.

29. The RRRM calculates the new parameters for the FDM Reconfigurable WMUX Algorithm, which determines the carrier frequency for each RAT, and then sends this information to the WMUX.

30. The WMUX receives the new parameters and stores them for use by the FDM Reconfigurable WMUX Algorithm.

31. After the default RAT was successfully constructed and a wireless connection established with the Access Network domain, the RCAL submits a request for new Reconfigurable Components and/or Algorithms from the Core Network to the Subscriber domain's SCM.

32. The Subscriber domain's SCM splits the message into a series of packets and transmits them to the Core Network domain's SCM using its Stop and Wait ARQ Reconfigurable Communications Algorithm and via the default RAT.

33. The Core Network domain's SCM receives the packets, reassembles the message, and forwards it to the Core Network domain's RCAL. The Core Network domain's RCAL determines that the Subscriber domain already possesses all available Reconfigurable Components and Algorithms, and therefore ignores the request.

34. The RPM submits a request for new policies from the Core Network to the Subscriber domain's SCM.

35. The Subscriber domain's SCM splits the message into a series of packets and transmits them to the Core Network domain's SCM using its Stop and Wait ARQ Reconfigurable Communications Algorithm and via the default RAT.

36. The Core Network domain's SCM receives the packets, reassembles the message, and forwards it to the Core Network domain's RPM. The Core Network domain's RPM determines that the Subscriber domain already possesses all available policies, and therefore ignores the request.

Once the Subscriber and Access Network domain's reconfigurable management systems have completed their initial operations and implemented their default 802.16 RATs, Network Layer packets can be exchanged between the two domains using the 802.16 RAT to modulate such packets onto signals for transmission through the wireless environment. The operations proceed as illustrated in Figure 5.18, and are described as follows:

1. The Access Network domain's SCM sends the first in a series of packets to the Access Network domain's Network Layer, for transmission to the Access Network domain's SCM via the default RAT.

2. The Access Network domain's Network Layer forwards this packet to the Access Network domain's PMUX, which stores the packet.

3. The Reconfigurable Component in the Access Network domain's default RAT, which is responsible for interfacing with the Access Network domain's PMUX, requests a packet to transmit from the Access Network domain's PMUX.

4. The Header Routing Reconfigurable PMUX Algorithm checks the requesting Reconfigurable Component's address against the routing parameters supplied by the Access Network domain's RRRM, and determines that all traffic must be routed via the default RAT.

5. The Access Network domain's PMUX sends the requesting Reconfigurable Component a packet for transmission. In the event that a reconfigurable management system packet was not available, a random packet would have been generated and sent.

6. The Access Network domain's default RAT's Reconfigurable Components process the supplied packet and produce a waveform to be transmitted. Until the waveform requested is requested by the Access Network domain's WMUX, it

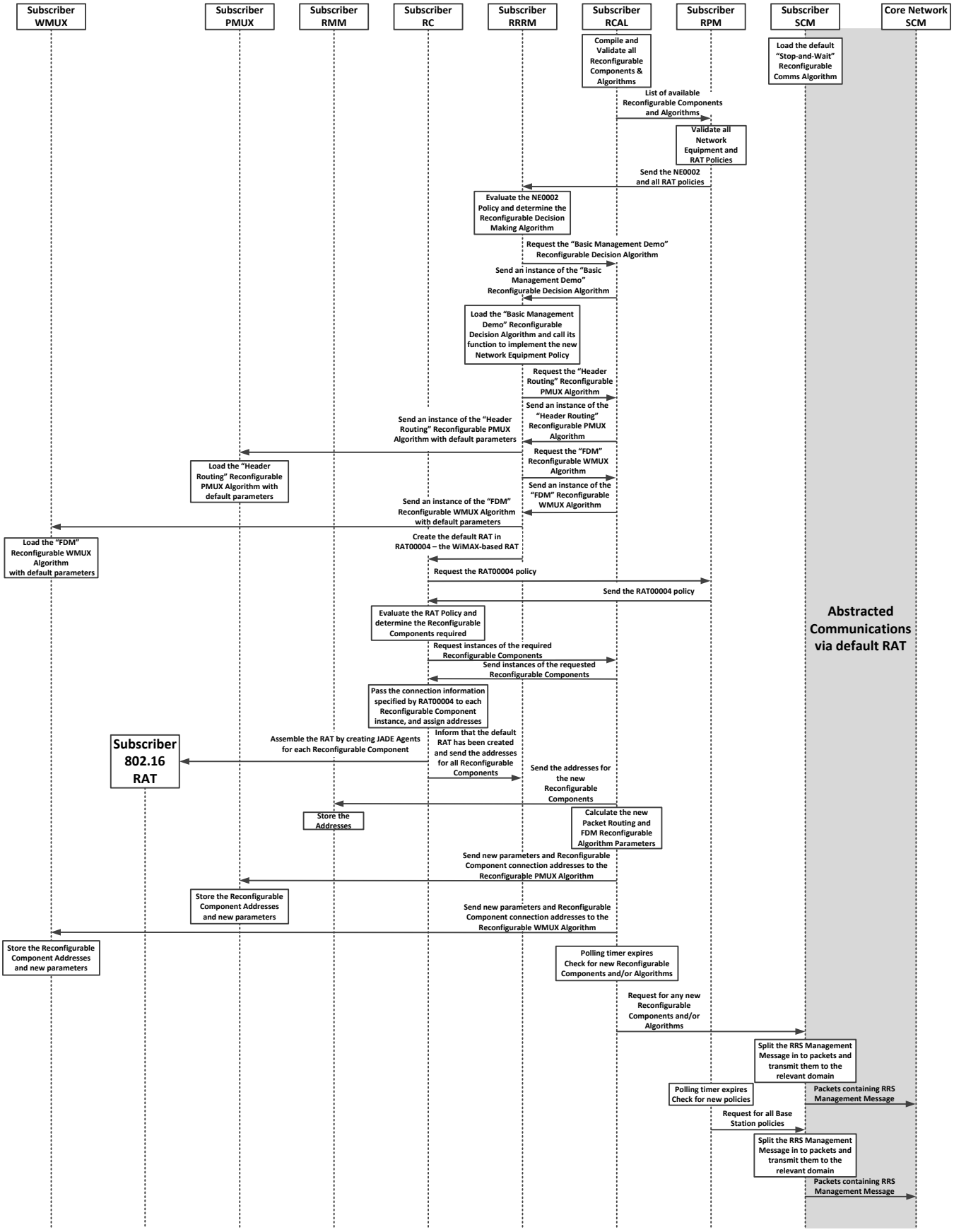


Figure 5.17: Reconfigurable Management System operations performed in the Subscriber domain during the initial phases of the CRRSA implementation simulation

is stored by the last Reconfigurable Component in the RAT which is connected to the Access Network domain's WMUX.

7. The Access Network domain's WMUX checks its Reconfigurable Component addresses supplied by the Access Network domain's RRRM, and sends a request to those Reconfigurable Components for a new waveform to transmit.

8. The Reconfigurable Component in the Access Network domain's default RAT sends the transmitting waveform to the Access Network domain's WMUX.

9. The Access Network domain's WMUX buffers the waveform and waits for the SDR API to request a new waveform.

10. The SDR API requests a new waveform to transmit from the Access Network domain's WMUX.

11. The Access Network domain's WMUX multiplexes and pulse-shapes the buffer RAT waveform(s) into a single waveform, prepends a synchronisation symbol, and sends the waveform to the SDR API.

12. As this is a No Wireless Channel Scenario, the SDR API simply copies the received waveform into its downlink receive buffer. Has this been either a MATLAB Simulated Wireless Channel or Real Wireless Channel Scenario, the waveform would have first been passed through a MATLAB function to simulate the channel, or transmitted between the USPR2s, before being stored in the downlink receive buffer. In addition, had this been a Real Wireless Channel Scenario, the wireless environment measurements taken by the USRP2s would have been saved by the SDR API.

13. The Subscriber domain's WMUX sends a request to the SDR API for received waveforms.

14. The SDR API sends the waveform stored in its downlink receive buffer.

15. The Subscriber domain's WMUX synchronises and de-multiplexes the received waveform resulting in each RAT's received waveform.

16. The Reconfigurable Component in the Subscriber domain's default RAT which is connected to the Subscriber domain's WMUX sends a request for a received waveform.

17. The Subscriber domain's WMUX responds with the requested waveform.

18. The Subscriber domain's default RAT's Reconfigurable Components process the received waveform and reproduce the transmitted packet. If this packet has been corrupted it is discarded, otherwise it is sent to the Subscriber domain's PMUX.

19. The Subscriber domain's PMUX checks the packet to determine if it is a random Network Layer packet or a reconfigurable management system packet. As this simulation sent a reconfigurable management system packet, it is sent to the Subscriber domain's Network Layer which forwards it to the Subscriber domain's SCM. Had it been a random Network Layer packet, it would have been counted and then discarded.

20. The Subscriber domain's SCM receives the packet, stores it, and transmits an ACK packet back to the Subscriber domain's Network Layer. This process then repeats itself in reverse.

During the course of these operations, the Reconfigurable Components in either domain sent context information reports to the RMM. The RMM stores this information, after determining which RAT the information belongs to using the Reconfigurable Component addresses supplied by the RRRM. The Reconfigurable Components in either domain may also request SDR measurements taken of the wireless environment from their domain's WMUX. As this is a No Wireless Channel Scenario, the values returned are all zero.

During the course of the simulation, the Subscriber and Access Network domains' RRRMs continuously receive context information reports from their respective RMMs, with the reports information describing the status and performance of the default 802.16 RAT. The Basic Management Demo Reconfigurable Decision Making Algorithm in the Subscriber domain's RRRM examines and dismisses each report until it detects that the Subscriber domain's 802.16 RAT has received a predefined number of packets. The reconfiguration management operations carried after this event are illustrated in Figure 5.19, and are described as follows:

1. The Subscriber domain's RRRM receives a context information report from the RMM and determines that a predefined number of packets has been received by its default 802.16 RAT.

2. The Basic Management Demo Reconfigurable Decision Making Algorithm in the Subscriber domain's RRRM determines that the LTE RAT should be created.

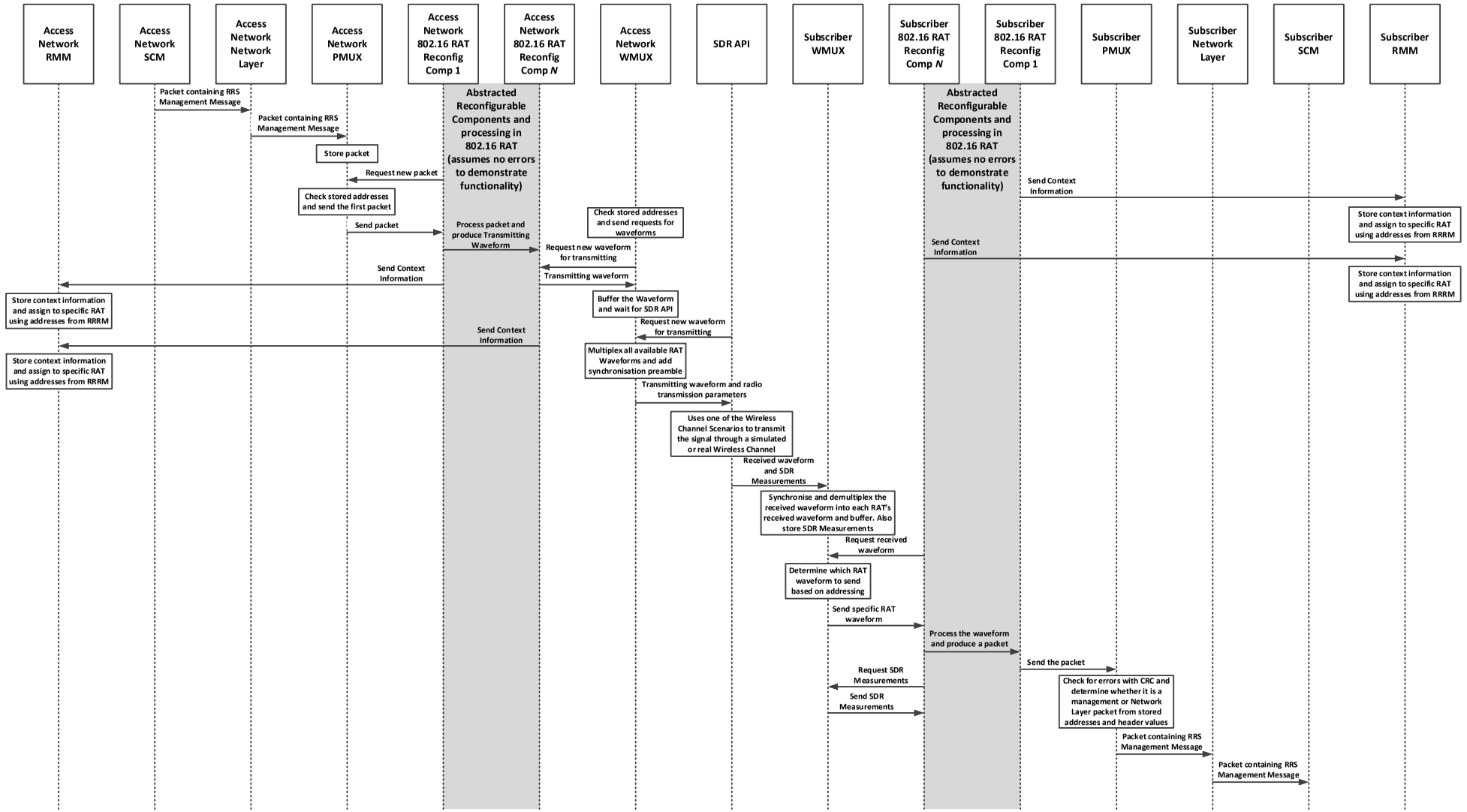


Figure 5.18: Operations performed reconfigurable management systems and RATs implemented in the Subscriber and Access Network domains during the CRRSA implementation simulation

3. The Subscriber domain's RRRM sends a reconfiguration request message to the Subscriber domain's SCM for transmission to the Access Network domain, specifying that it would like to create the LTE RAT.

4. The Subscriber domain's SCM splits the message into a series of packets and transmits them to the Access Network domain's SCM using its Stop and Wait ARQ Reconfigurable Communications Algorithm and via the default RAT.

5. The Access Network domain's SCM receives the packets, reassembles the message, and forwards the reconfiguration request to the Access Network domain's RRRM.

6. The Basic Management Demo Reconfigurable Decision Making Algorithm in the Access Network domain's RRRM determines that the request can be honoured and that the LTE RAT can be created.

7. The Access Network domain's RRRM sends a reconfiguration response message to the Access Network domain's SCM for transmission to the Subscriber domain, specifying that the creation of the LTE RAT may proceed.

8. The Access Network domain's SCM splits the message into a series of packets and transmits them to the Subscriber domain's SCM using its Stop and Wait ARQ Reconfigurable Communications Algorithm and via the default RAT.

9. The Subscriber domain's SCM receives the packets, reassembles the message, and forwards the reconfiguration response to the Subscriber domain's RRRM.

10. The Subscriber and Access Network domains' logical components both perform the following operations:

    (a) The RRRM instructs the RC to create the LTE RAT using the RAT Policy which contains the specification for the LTE RAT model. Note that the RAT00001 is used in the Access Network domain, and RAT00002 in the Subscriber domain.

    (b) The RC receives the instruction and sends a request to the RPM for a copy of the relevant RAT Policy.

    (c) The RPM sends the RC a copy of the requested RAT Policy.

    (d) The RC examines the RAT Policy, and requests instances of the specified Reconfigurable Components from the RCAL.

    (e) The RCAL responds with the requested Reconfigurable Component instances.

(f) The RC encapsulates the Reconfigurable Component instances in agents, assigns each agent a unique address, and provides the connection information needed by each agent to acquire inputs and distribute outputs.

(g) The RC informs the RRRM that the RAT has been created, and provides a list of all the Reconfigurable Component agent's addresses.

(h) The RRRM stores the addresses, and also sends them to the RMM.

(i) The RMM stores the received addresses, and uses the information to determine which Reconfigurable Components belong to each RAT. This is necessary as the RMM constantly receives context information from Reconfigurable Components, and each item of information must be attributed to a particular RAT so that when the RMM compiles its context information report for the RRRM, it knows which items of information to use for each RAT.

(j) The RRRM calculates the new parameters for the Header Routing Reconfigurable PMUX Algorithm, which determines how traffic is scheduled between the existing RATs, and then sends them to the PMUX.

(k) The PMUX receives the new parameters and stores them for use by the Header Routing Reconfigurable PMUX Algorithm.

(l) The RRRM calculates the new parameters for the FDM Reconfigurable WMUX Algorithm, which determines the carrier frequency for each RAT, and then sends this information to the WMUX.

(m) The WMUX receives the new parameters and stores them for use by the FDM Reconfigurable WMUX Algorithm.

The final phase of the simulation is initiated when the Basic Management Demo Reconfigurable Decision Making Algorithm in the Access Network domain's RRRM detects that the Access Network domain's LTE RAT has received a predefined number of packets. The reconfiguration management operations carried after this event are illustrated in Figure 5.20, and are described as follows:

1. The Access Network domain's RRRM receives a context information report from the RMM and determines that a predefined number of packets has been received by its LTE RAT.

2. The Basic Management Demo Reconfigurable Decision Making Algorithm in the Access Network domain's RRRM determines that the 802.16 RAT should be terminated.

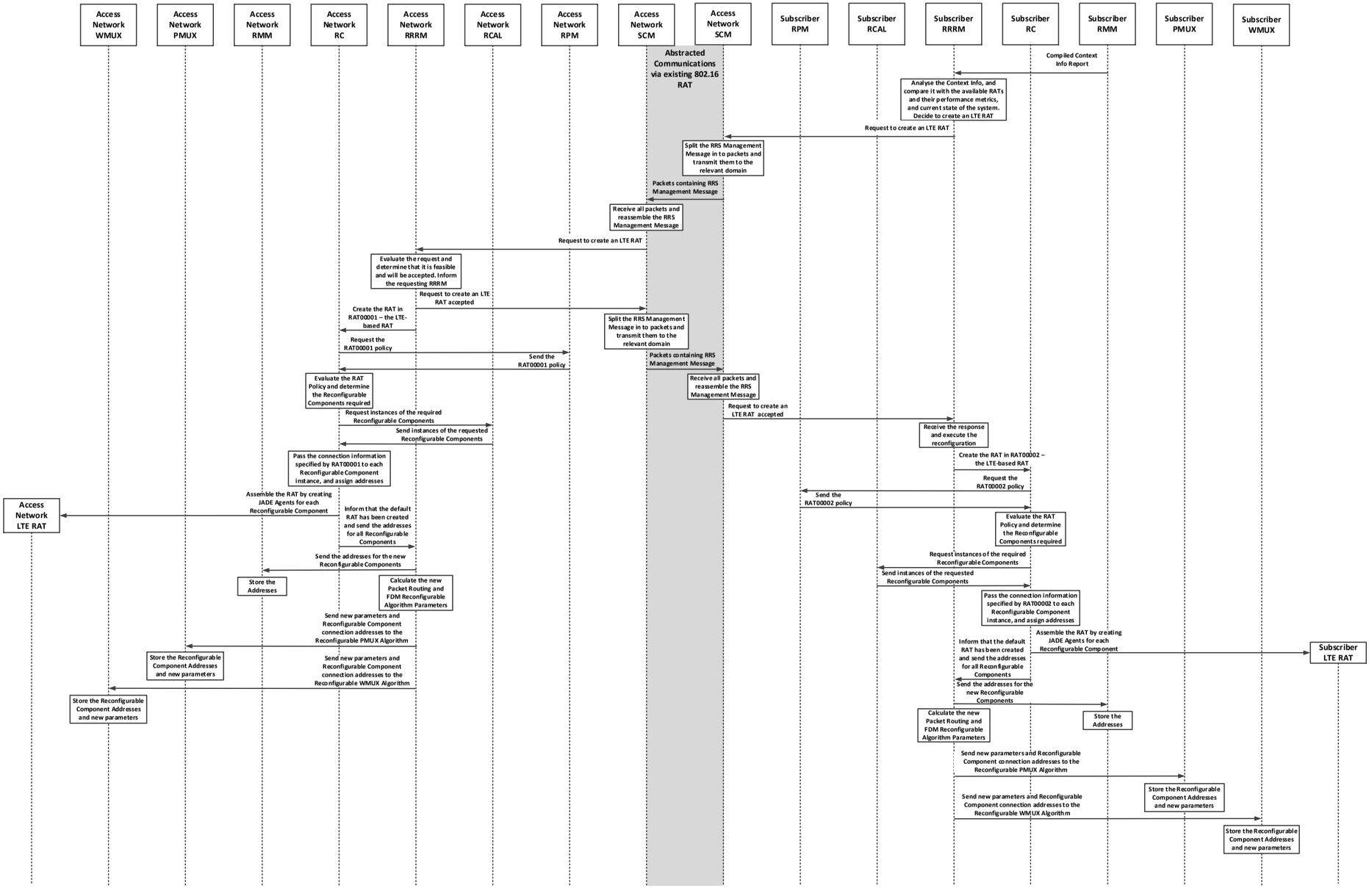


Figure 5.19: Reconfigurable Management System operations performed in the Subscriber and Access Network domains to create the LTE RAT during the CRRSA implementation simulation

3. The Access Network domain's RRRM sends a reconfiguration request message to the Access Network domain's SCM for transmission to the Subscriber domain, specifying that it would like to terminate the 802.16 RAT.

4. The Access Network domain's SCM splits the message into a series of packets and transmits them to the Subscriber domain's SCM using its Stop and Wait ARQ Reconfigurable Communications Algorithm and via the 802.16 and LTE RAT.

5. The Subscriber domain's SCM receives the packets, reassembles the message, and forwards the reconfiguration request to the Subscriber domain's RRRM.

6. The Basic Management Demo Reconfigurable Decision Making Algorithm in the Subscriber domain's RRRM determines that the request can be honoured and that the 802.16 RAT can be terminated.

7. The Subscriber domain's RRRM sends a reconfiguration response message to the Subscriber domain's SCM for transmission to the Access Network domain, specifying that the termination of the 802.16 RAT may proceed.

8. The Subscriber domain's SCM splits the message into a series of packets and transmits them to the Access Network domain's SCM using its Stop and Wait ARQ Reconfigurable Communications Algorithm and via the 802.16 and LTE RAT.

9. The Access Network domain's SCM receives the packets, reassembles the message, and forwards the reconfiguration response to the Access Network domain's RRRM.

10. The Subscriber and Access Network domains' logical components both perform the following operations:

    (a) The RRRM calculates the new parameters for the Header Routing Reconfigurable PMUX Algorithm, which determines how traffic should be rescheduled so that the 802.16 RAT stops receiving new Network Layer packets.

    (b) The PMUX receives the new parameters and stores them for use by the Header Routing Reconfigurable PMUX Algorithm.

    (c) The RRRM calculates the new parameters for the FDM Reconfigurable WMUX Algorithm, which determines the carrier frequency for the remaining LTE RAT, and then sends this information to the WMUX.

(d) The WMUX receives the new parameters and stores them for use by the FDM Reconfigurable WMUX Algorithm.

(e) The RRRM instructs the RC to terminate the 802.16 RAT.

(f) The RC receives the instruction and instructs each Reconfigurable Component in the 802.16 RAT to complete its current task and then terminate itself.

(g) The RC informs the RRRM that the RAT has been terminated.

(h) The RRRM informs the RMM that the RAT has been terminated.

(i) The RMM removes the terminated RAT's Reconfigurable Component's addresses.

The operations performed during the No Wireless Channel Scenario simulation, which are presented in this section, demonstrate all of the functionality provided by the CRRSA's reconfigurable management system and operating environment. The following section demonstrates that the 802.16 and LTE RATs are capable of successfully operating in a simulated and real wireless environment.

5.4.2 RAT Functionality Results

A key objective of this chapter is to demonstrate that RAT models developed using the CRRSA are capable of successfully transmitting a signal between domains without errors, and that multiple RATs can be simultaneously implemented without compromising any of the RATs' performance. To achieve this, two additional sets of simulations were performed using the MATLAB Simulated Wireless Channel Scenario and the Real Wireless Channel Scenario, and the same series of reconfiguration management operations that were described in the previous section. The default parameters used for both sets of simulations, as well as the Network Equipment and RAT Policies, the Reconfigurable Components, and Reconfigurable Algorithms, were all as described in the previous sections of this chapter.

It is important to note at the outset that the purpose of performing these simulations was to demonstrate that the required RATs were implemented as described in this chapter and Chapter 4, and that they were able to successfully transmit Network Layer packets between domains in both a simulated and real wireless environment. The purpose was not to measure or analyse the performance of the 802.16 and LTE RATs in either a simulated or real wireless environment. To demonstrate the

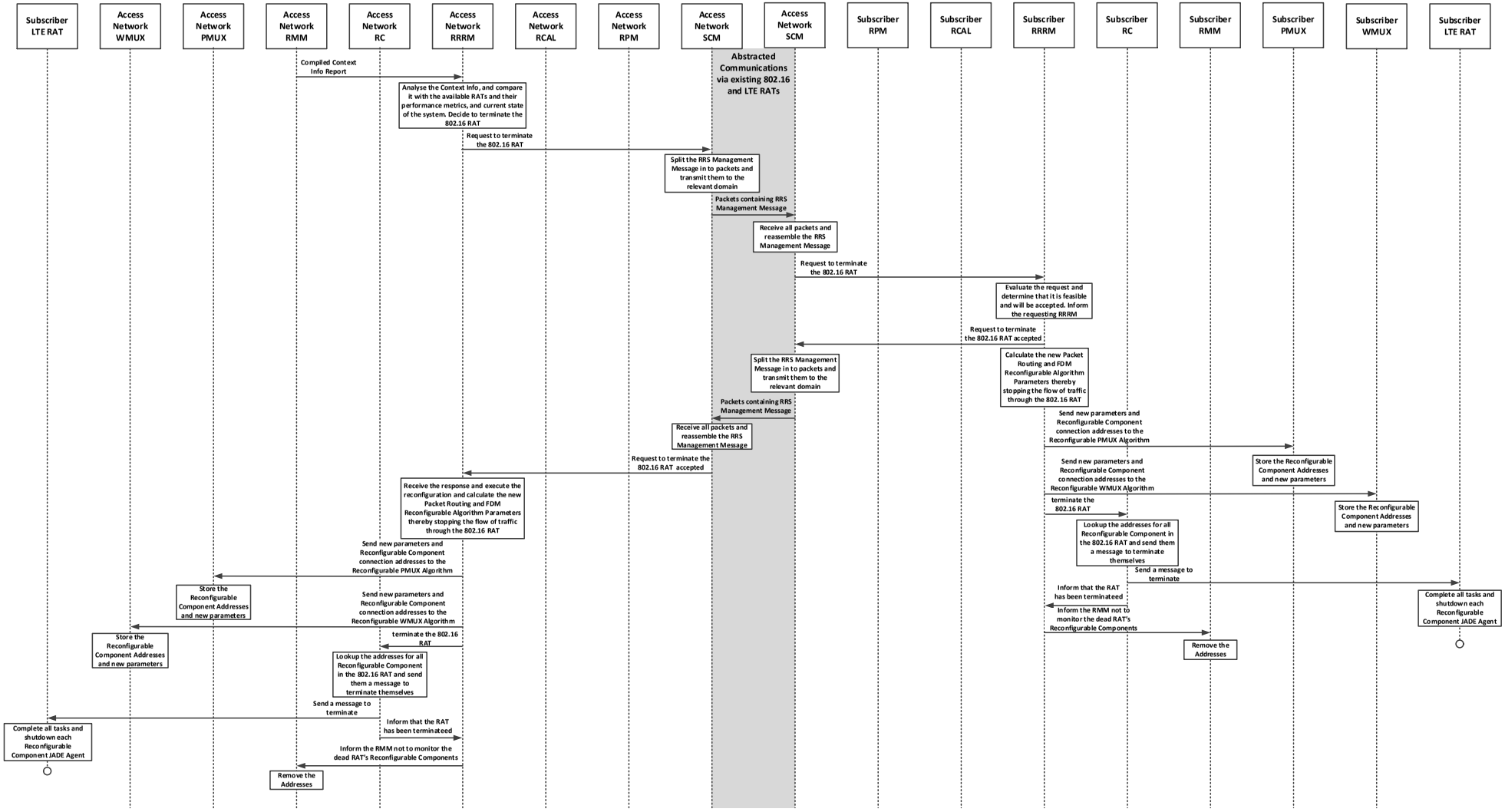


Figure 5.20: Reconfigurable Management System operations performed in the Subscriber and Access Network domains to terminate the 802.16 RAT during the CRRSA implementation simulation

success of the simulations, measurements were taken from specific Reconfigurable Components in the 802.16 and LTE RATs during each simulation.

The first set of such measurements taken is presented in Figure 5.21, which contains the constellation diagrams of all transmitted, received, and equalised symbols in the 802.16 and LTE RATs during the MATLAB Simulated Wireless Channel and the Real Wireless Channel Scenarios. The purpose of these diagrams is to demonstrate the amount of interference and phase noise introduced by each Wireless Channel Scenario, and how the Least Squares Channel Estimation and Zero Forcing Equalisation Reconfigurable Components are able to compensate for the interference and phase noise and ensure that the symbols can be correctly demodulated.

The signals presented in Figure 5.22 contain the time domain representations of the signals transmitted and received by the 802.16 and LTE RATs in each Wireless Channel Scenario. The transmitted signals clearly illustrate the OFDM synchronisation preamble which is generated and prepended by the FDM Reconfigurable WMUX Algorithm, while the received signals show the noise introduced by each of the Wireless Channel Scenarios as well as the time delay which the FDM Reconfigurable WMUX Algorithm must estimate using the synchronisation preamble.

Figure 5.23 presents the frequency domain representations of the signals transmitted and received by the 802.16 and LTE RATs in each Wireless Channel Scenario, and shows each RAT's transmission bandwidth after pulse shaping, and the noise introduced by each Wireless Channel Scenario.

Figure 5.24 illustrates the 802.16 and LTE RATs' OFDM subcarriers in the frequency domain before and after transmission through each of the Wireless Channel Scenarios. The purpose of these diagrams is to show the format, bandwidth, and subcarrier spacing of each OFDM symbol, and the frequency selective fading and noise introduced by each Wireless Channel Scenario.

Finally, Figure 5.25 demonstrates how multiple RAT waveforms were pulse-shaped and multiplexed using the FDM Reconfigurable WMUX Algorithm while both RATs were operating simultaneously without introducing intersymbol interference.

5.4.3 Critical Analysis of the CRRSA Implementation

The objectives for the implementation of the CRRSA were presented in Section 5.1, and formed the basis for the design and development decisions made throughout

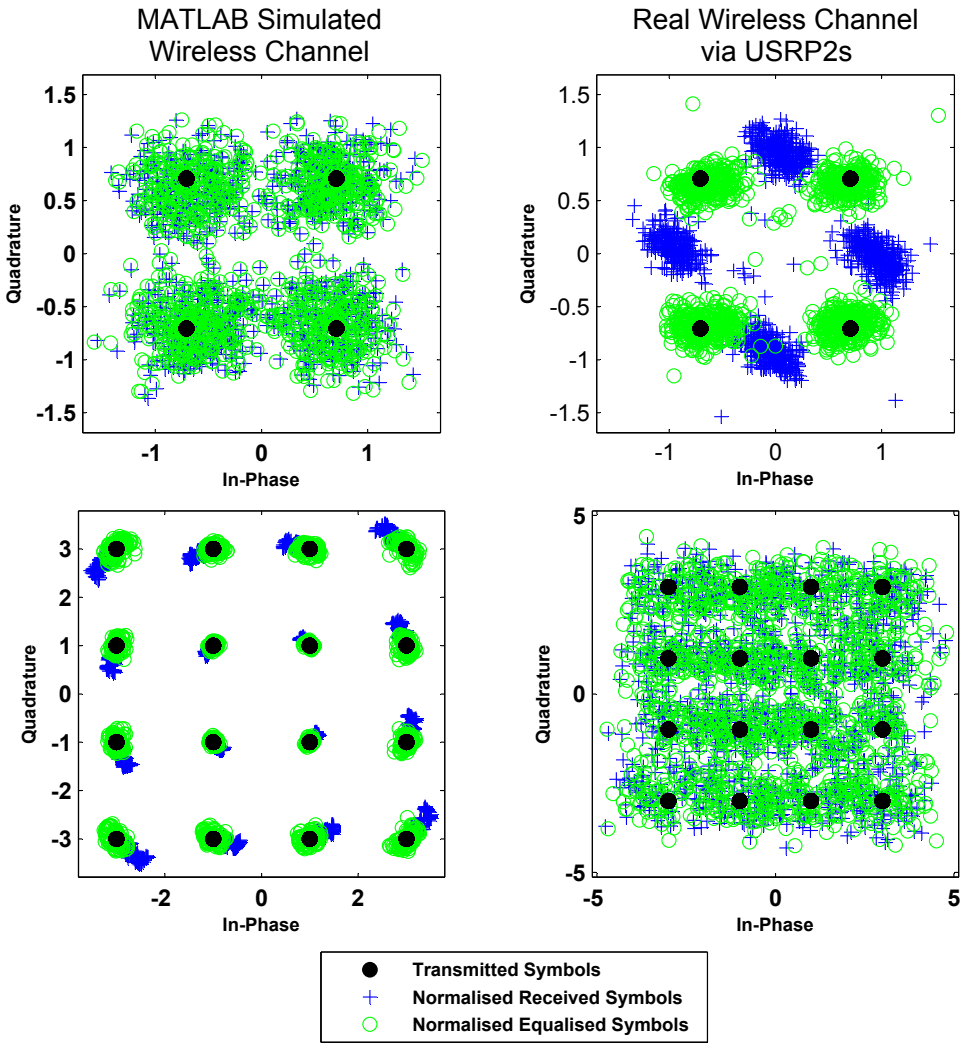


Figure 5.21: Constellation diagrams containing transmitted, received, and equalised symbols from the 802.16 and LTE RATs implemented during the MATLAB Simulated Wireless Channel and the Real Wireless Channel Scenarios

the rest of the chapter involving the implementation of each component within the CRRSA, and determining which parameters and scenarios are used to test its functionality.

The execution of the implementation was tested in the previous sections in a set of simulations designed to demonstrate all of the functionality provided by the CRRSA's reconfiguration management system, operating environment, and RAT modelling and implementation process. The results of these simulations must now be analysed to determine if in fact the implementation does meet its predefined objectives. This analysis is presented in Table 5.9, where all the implementation

objectives are shown to have been fulfilled by specific phases of the simulations depicted in the previous two sections. As such the CRRSA implementation can be considered a success.

5.5 Summary

The purpose of this chapter was to prove that the CRRSA presented in Chapter 3 is capable of meeting all of the functional requirements defined in Chapter 2, and that it is also able to successfully implement the RAT models developed in Chapter 4. To achieve these goals, a series of implementation objectives were defined, and the specific technologies used to perform the implementation were identified and described in detail.

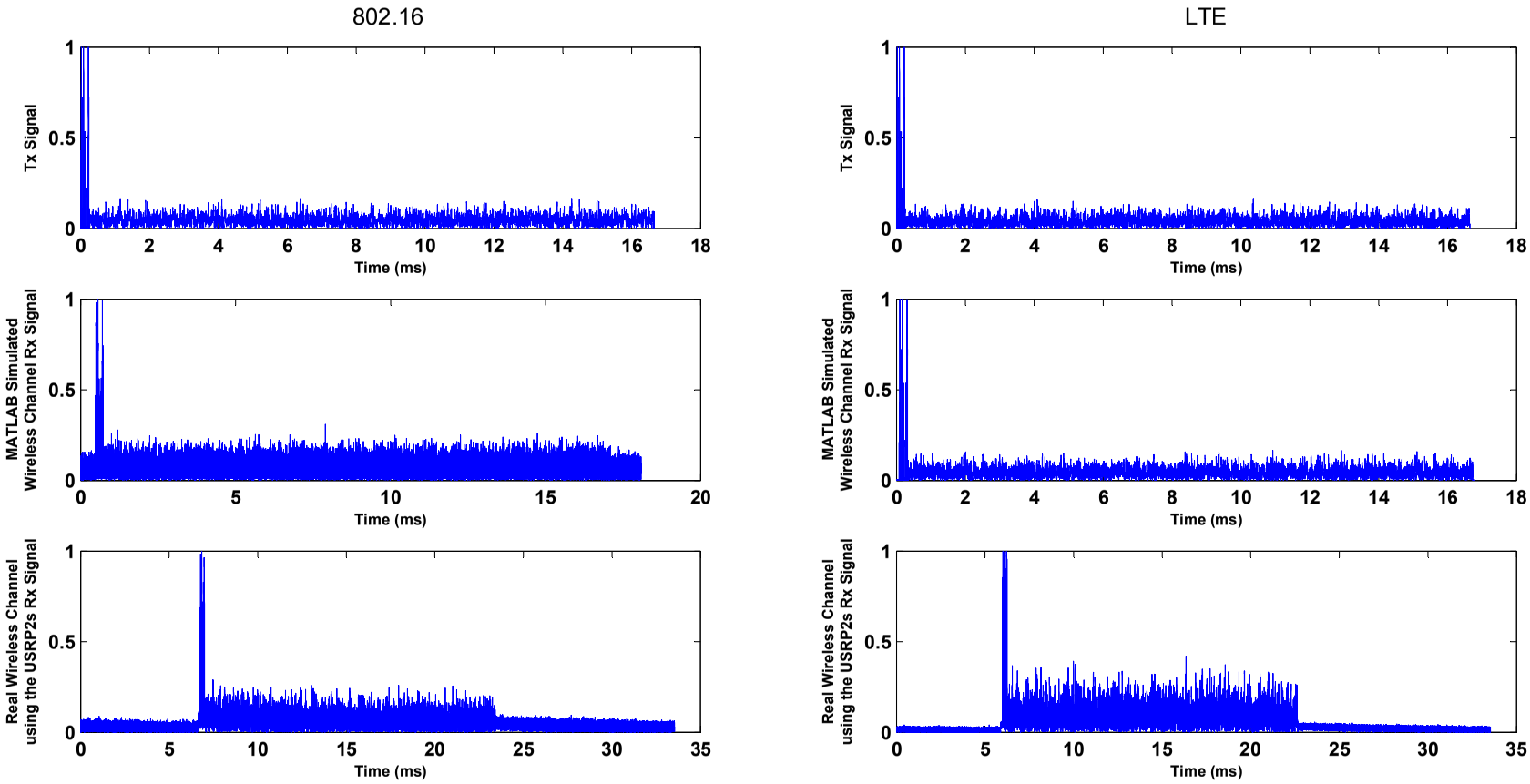


Figure 5.22: Signals Transmitted and Received by the 802.16 and LTE RATs implemented during the MATLAB Simulated Wireless Channel and the Real Wireless Channel Scenarios

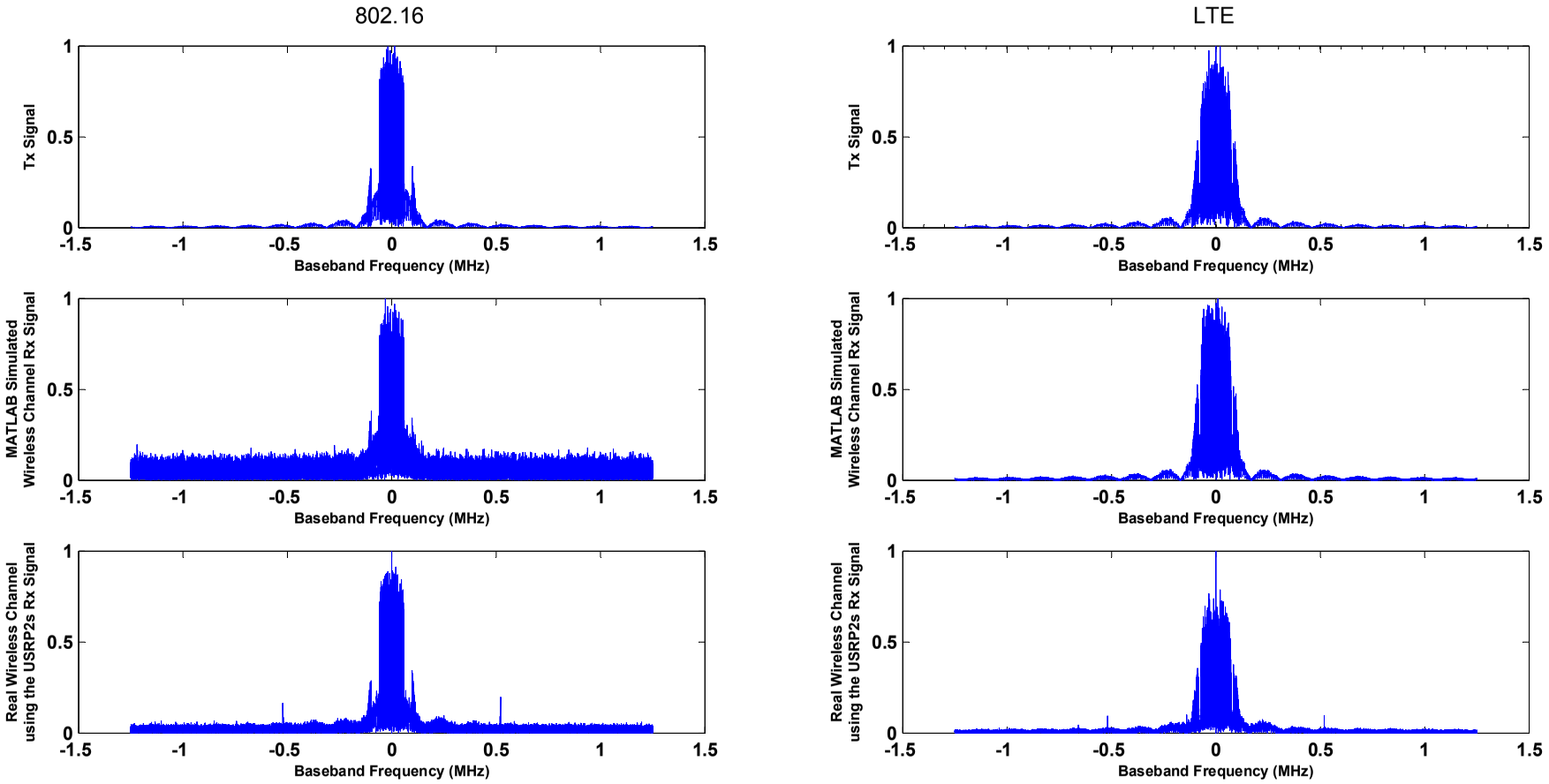


Figure 5.23: Frequency domain representations of signals Transmitted and Received by the 802.16 and LTE RATs implemented during the MATLAB Simulated Wireless Channel and the Real Wireless Channel Scenarios

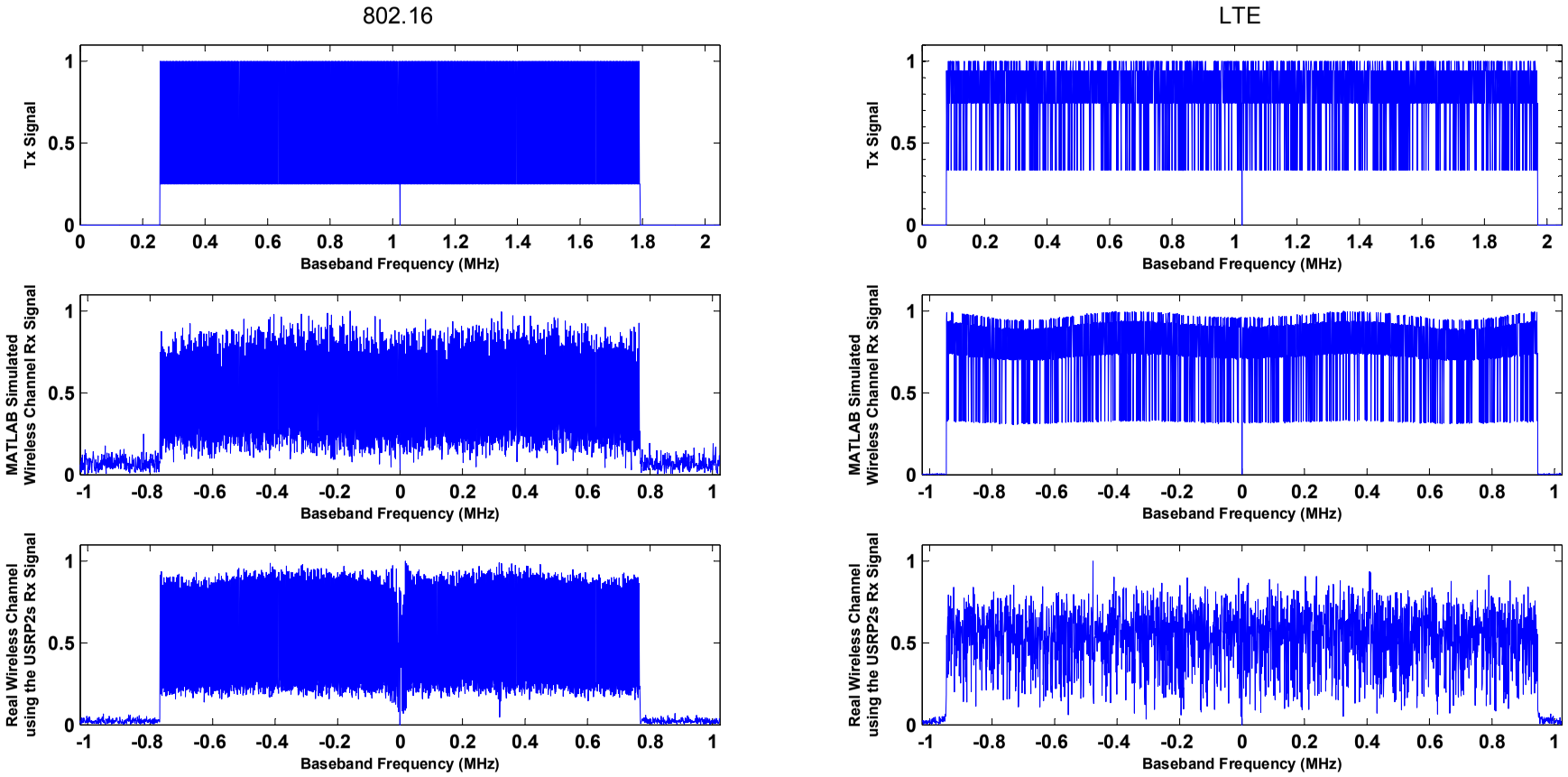


Figure 5.24: OFDM Subcarriers of the 802.16 and LTE RATs implemented during the MATLAB Simulated Wireless Channel and the Real Wireless Channel Scenarios

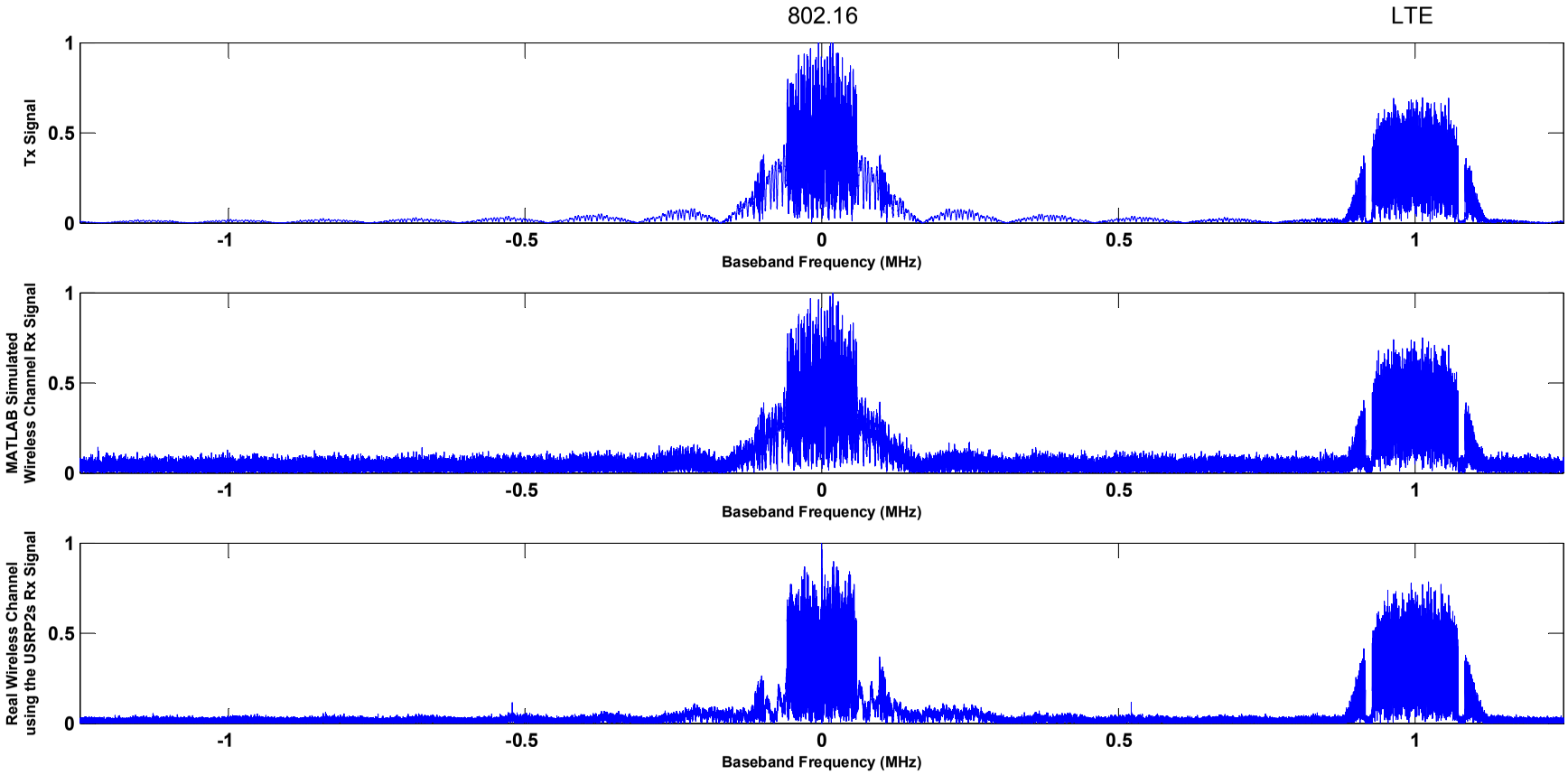


Figure 5.25: Frequency domain representations of signals Transmitted and Received by the 802.16 and LTE RATs implemented simultaneously during the MATLAB Simulated Wireless Channel and the Real Wireless Channel Scenarios

Table 5.9: Correlating the objectives of the CRRSA implementation with the results of the simulations conducted to test the functionality of the implementation

Implementation Objective	**How was the objective fulfilled/demonstrated?**
Reconfigurable Management Systems must independently and autonomously launch and initialise their logical components using parameters defined in policies	Demonstrated at the start of the No Wireless Channel Scenario Simulation and presented in Figures 5.15, 5.16, and 5.17
The Subscriber and Access Network domains must automatically launch their default RATs and establish and maintain a wireless connection based on information defined in policies	Demonstrated at the start of the No Wireless Channel Scenario Simulation and presented in Figures 5.15, 5.16, 5.17, and 5.18
Reconfigurable Management Systems must be able to dynamically modify their radio resource and traffic distribution, spectrum allocation, transmission parameters, and inter-domain communications protocols	Demonstrated as part of the reconfiguration operations performed in the No Wireless Channel Scenario Simulation and presented in Figures 5.19 and 5.20
Reconfigurable Management Systems must be able to make and execute reconfiguration decisions at runtime based on context information and policies	Demonstrated as part of the reconfiguration operations performed in the No Wireless Channel Scenario Simulation and presented in Figures 5.19 and 5.20
New policies, Reconfigurable Components and Algorithms must be automatically downloaded, validated and, if necessary, executed	Demonstrated at the start of the No Wireless Channel Scenario Simulation and presented in Figures 5.15, 5.16, and 5.17
Multiple RATs must be able to be implemented simultaneously	Demonstrated as part of all Wireless Channel Scenario Simulations. The reconfiguration operations which launch a second RAT are presented in Figure 5.19, and Figure 5.25 illustrates how multiple RAT waveforms are multiplexed into a single waveform for transmission
Implemented RATs must be capable of successful transmission, i.e. transmitting and receiving signals carrying data without errors	Demonstrated as part of all Wireless Channel Scenario Simulations. Figure 5.18 presented the general RAT transmission operations carried out by Reconfigurable Components and various components in the Reconfigurable Management systems, and Figures 5.21-5.25 depicted the measurements taken from the 802.16 and LTE RATs during simulations which demonstrate the functionality and signal processing operations provided by the implemented Reconfigurable Components and RATs

Chapter 6

Conclusion

Mobile Communications Systems have become a pervasive part of modern life, resulting in an ever increasing demand for fast ubiquitous access to multimedia and data services. This demand has resulted in the need for telecommunications standards bodies to continuously upgrade and develop new RATs with more advanced signal processing techniques, and radio resource management systems

As new RATs are developed, mobile network operators are usually forced to purchase new hardware systems capable of supporting the new RATs' functionality, and then to integrate these systems into their existing heterogeneous RANs. This continuous evolution in RAT technology places a material burden on network operators, who must not only spend significant amounts of their capital on purchasing new network equipment, but must also convince subscribers to purchase new mobile devices which support the new RATs in order to obtain a return on their network investment.

Reconfigurable Radio Systems offer an alternative and more flexible method of standardising, developing, deploying, and managing RATs. By defining each RAT's functionality in software which can be reused across multiple hardware platforms, and by providing standardised reconfigurable management systems and operating environments, a network operator is able to dynamically reconfigure network elements to use different RATs and thereby minimise costly network and terminal upgrades. This reconfigurable functionality can be further exploited to jointly manage the radio resources of a heterogeneous RAN, and to enable the dynamic and more efficient allocation of spectrum.

Although many researchers and standardisation initiatives have developed their own RRS architectures or frameworks, in full or in part, there is no unified approach or comprehensive set of functional requirements for developing RRSs. This lack

of a standardised approach to developing RRSs is a significant hindrance to the widespread adoption and implementation of RSSs in RANs.

The purpose of this thesis was to use a greenfield approach in order to design a logical architecture, which can in turn be used to develop and implement a RRS which combines and improves on the functionality of all the major RRSs developed by researchers and standards initiatives, and which also eliminates or avoids their various shortcomings. The architecture, termed the Converged RRS Architecture, is capable of dynamically implementing any legacy, current and future RAT at runtime, and it also provides a flexible reconfigurable management system which can be upgraded and customised to suit a network operation's needs.

The following sections summarise the work presented in this thesis, highlight the key contributions made to the RRS field of research, and present a series of topics for future research.

6.1 Functional Requirements for the Converged RRS Architecture

The Converged RRS Architecture (CRRSA) was developed using a set of functional requirements which were identified and confirmed by reviewing and analysing all major RRSs developed by researchers and standardisation initiatives. These requirements were divided into three categories based on the types of functions they specify, and are summarised below:

- **Operating Environment Functional Requirements**: The CRRSA must provide a secure operating environment to support the reconfiguration management operations which are carried out by the reconfigurable management system in each domain. This operating environment must provide a standardised interface between RATs and SDR hardware platforms, and a secure communications system for reconfiguration management messages to be passed between domains.
- **Reconfigurable Management System Functional Requirements**: The CRRSA must provide a reconfigurable management system in each domain which is capable of making and executing reconfiguration decisions at runtime, based on context information obtained from existing RATs, a list of available

RATs, and predefined radio resource management and spectrum allocation objectives and limitations. The reconfigurable management system is also responsible for providing a mechanism to download and validate new policies, software components and radio applications, and a Cognitive Pilot Channel solution to enable mobile devices to connect to base stations when they are first booted up.

- **Radio Application Functional Requirements**: The CRRSA must provide a radio application development framework to assist developers in developing RATs which are portable and interoperable with separate CRRSA implementations. This development framework must be able to implement the Reconfigurable Protocol Stack approach to developing RATs, in order to ensure that the CRRSA is capable of implementing any legacy, current, or future RAT.

6.2 The Converged RRS Architecture

The Converged RRS Architecture is a technology independent logical architecture which contains abstracted logical components, standardised interfaces, and predefined operating procedures. Each logical component is responsible for implementing at least one of the CRRSA's functional requirements in network elements contained within the Subscriber, Access Network, or Core Network domains.

To enable the CRRSA's reconfigurable management system to be customised according to deployment scenarios, or to be upgraded to accommodate new functionality and/or technologies, the CRRA defines a customisable logical component called a Reconfigurable Algorithm. There are four different types of Reconfigurable Algorithms supported by the CRRSA, which determine how reconfiguration management messages are securely exchanged between domains, traffic is scheduled between RATs, RAT waveforms are combined for transmission, and reconfiguration decisions are made.

Network Equipment policies enable the CRRSA's reconfigurable management systems to perform their functions autonomously at runtime, by defining which Reconfigurable Algorithm implementations are to be used in each domain, and which RATs may be implemented through reconfiguration operations. These types of policies also specify which RAT must be launched whenever a mobile device or base station is booted, thereby providing a CPC solution.

The CRRSA implements RATs using logical building blocks called Reconfigurable Components, which are connected via standardised interfaces. Each Reconfigurable Component encapsulates a single atomic function which can be reused to construct the Physical and Data Link Layers of different RATs. The CRRSA provides a standardised interface between RATs and SDR hardware platforms to enable Reconfigurable Components to be reused in different CRRSA implementations. During normal transmission operations, Reconfigurable Components must submit context information reports to their reconfigurable management system, which provides the system with the information it needs to determine whether and when a reconfiguration operation should be performed. RATs are implemented at runtime using RAT policies to determine which Reconfigurable Components are to be utilised, and how they should be connected. Each RAT policy also contains performance metrics for its RAT which is provided to the CRRSA's reconfiguration management system to use during its reconfiguration decision making process.

Finally, the Core Network's reconfiguration management system provides the necessary functionality to ensure that all network elements in the Subscriber and Access Network domains are kept up to date regarding available Reconfigurable Components, Algorithms and policies by deploying them to such network elements whenever new applicable versions are detected.

6.3 Implementing the Converged RRS Architecture

In order to prove that the CRRSA is capable of performing all necessary reconfiguration management operations, and that it can also successfully implement RATs, the CRRSA was implemented and tested via a series of simulation scenarios using two RAT models derived from the IEEE 802.16 and 3GPP LTE standards.

These simulations tested the functionality of the CRRSA and demonstrated that its logical components possess the ability to initialise each domain's reconfiguration management system using Reconfigurable Algorithms and information contained in Network Equipment policies, and then to successfully implement the Reconfigurable Components required to construct the 802.16 RAT, which had been selected as the default RAT for simulation purposes.

The initial stages of these simulations demonstrated that the 802.16 default RAT was able to successfully transmit and receive RF signals carrying data without errors in

both a simulated and a real wireless environment. Once the 802.16 default RAT had received a predefined number of packets, the CRRA implementation demonstrated its capability to perform dynamic reconfigurations at runtime, by first constructing a second LTE RAT, and then terminating the original 802.16 RAT. The LTE RAT then continued to successfully transmit and receive the RF signals without errors.

These simulations thus successfully proved that the CRRSA is capable of implementing its reconfiguration management and RAT functionality in a practical wireless environment.

6.4 Key Contributions

The primary purpose of this thesis is to develop a logical architecture, based on a greenfield approach, which combines the required functionality of all major existing RRSs into a single system, and which is capable of autonomously managing and controlling the secure dynamic reconfiguration of RATs in a RAN at runtime. Further, this logical architecture must be demonstrated to be flexible, scalable, practically realisable, and able to implement any legacy, current or future RATs.

The Converged Reconfigurable Radio System Architecture is a unique logical architecture which was developed to meet these objectives. The CRRSA is an extremely flexible and scalable architecture which defines a set of technology-independent logical components and interfaces for managing and performing reconfiguration operations in the Subscriber, Access Network and Core Network domains.

Unlike most existing RRSs, the CRRSA provides a flexible, scalable method for implementing RATs in software and interfacing such RATs with a generic SDR hardware platform, as well as a reconfiguration management system and operating environment for managing and controlling the dynamic reconfiguration of RATs at runtime. Although some RRSs do refer to some of these elements conceptually in their documentation, they do not provide detailed specifications for their implementation. The CRRSA on the other hand provides extremely detailed specifications for the operations performed by every logical component, the messages exchanged between such components, and the overall reconfiguration operations carried out by the architecture. The CRRSA is also unique, in that it provides the capability to customise and upgrade its reconfiguration management system through the use of Reconfigurable Algorithms, and to do so dynamically and with complete flexibility.

The CRRSA was also demonstrated to be capable of practically implementing two RATs, based on modern RAT standards, and using Reconfigurable Components, which are not only able to successfully transmit signals in a simulated environment, but are also able to transmit and receive signals carrying data through the wireless environment using an actual SDR hardware platform. This is also a key differentiator when comparing the CRRSA with the work performed by other researchers in the field, as they have not demonstrated how to practically implement and integrate reconfigurable management systems and software defined RATs using SDR hardware. Instead, most researchers either simulate their reconfigurable management systems, or demonstrate that they can implement RATs using SDR hardware platforms, but no researchers have covered both to any material extent. The key contributions made by the CRRSA to the field of RRS research can therefore be summarised as follows:

1. The CRRSA is a logical architecture, based on a greenfield approach, which includes the functionality provided by all major existing RRSs, and which provides a detailed unified approach to the development and standardisation of RRSs. The design approach adopted avoids the limitations of existing RRSs, which cannot be modified or upgraded to provide the same level of functionality, both for practical reasons and due to architectural and technological limitations.

2. The CRRSA provides extremely flexible and scalable methods for reconfiguring the functionality provided by its reconfigurable management system, as well as for reconfiguring the RATs themselves.

3. The CRRSA demonstrates that reconfigurable management systems and software defined RATs can be integrated into a single system which is practically realisable and capable of operating in a real wireless environment using SDR hardware platforms.

6.5 Future Work

During the course of the work presented in this thesis, a number of areas were identified as potential topics for future RRS research and implementation of the CRRSA. These are summarised as follows:

- The CRRSA abstracts all Core Network functions which are not directly related to managing and performing reconfiguration operations in the Subscriber and Access Network domains. A potential topic for future research would be to investigate which elements in the Core Network could be used to enhance the functionality already provided by the CRRSA, and to develop standardised interfaces to enable such network elements to interface with the logical components contained in the CRRSA's reconfigurable management system.

- As the CRRSA implementation necessarily focused on proving the functionality of the CRRSA and its capability to successfully implement RATs in a link level simulation, several types of Data Link Layer related functions were not modelled and implemented in Reconfigurable Components. These functions could be modelled and implemented in a system level simulation, to expand the number of Reconfigurable Components available for implementing RATs and to demonstrate how reconfigurable network elements would interact. Examples of such functions include mobility management, radio link control, and dynamic management of the RATs' radio resources.

- Due to the limitations of the SDR hardware platform used to implement the CRRSA, several advanced digital signal processing techniques could not be implemented. A potential topic for future work could involve the implementation of these digital signal processing techniques in Reconfigurable Components in different SDR hardware platforms, and testing them in new RAT models implemented on different SDR hardware platforms.

- The Reconfigurable Decision Making Algorithm developed for the CRRSA implementation performed reconfiguration operations based on a series of predefined instructions. Many researchers in the field of Cognitive Radio have investigated the use of artificial intelligence algorithms capable of learning from previous reconfiguration decisions, thereby improving the efficiency of radio resource and spectrum allocation, and providing the RATs chosen to support mobile services with specific QoS requirements.

- The digital signal processing techniques used in mobile communication systems are substantially more complex than those used in fixed line communications. An interesting subject for future research would be to determine how the functionality provided by fixed line communication systems could be implemented in software, thereby enabling such systems to be reconfigured to implement new fixed line communication standards based on generic hardware platforms.

References

[1] J. Gillet. "Africa Mobile Connections Q3 2011." Tech. rep., Wireless Intelligence, November 2011.

[2] R. Prasad and A. Mihovska, editors. *New Horizons in Mobile and Wireles Communications: Reconfigurability*. Mobile Communication Series. Artech House, 2009.

[3] T. Ulversoy. "Software Defined Radio: Challenges and Opportunities." *Communications Surveys Tutorials, IEEE*, vol. 12, no. 4, pp. 531 –550, quarter 2010.

[4] IEEE Standards Coordinating Committee 41 on Dynamic Spectrum Access Networks. "IEEE Std 1900.1: IEEE Standard Definitions and Concepts for Dynamic Spectrum Access: Terminology Relating to Emerging Wireless Networks, System Functionality, and Spectrum Management.", 2008.

[5] R. Tafazolli, editor. *Technologies for the Wireless Future*, vol. 2. John Wiley & Sons, Ltd., 2006.

[6] H. Arslan, editor. *Cognitive Radio, Software Defined Radio, and Adaptive Wireless Systems*. Springer, 2007.

[7] ETSI Technical Committee for RRSs. "TR 102-838: Reconfigurable Radio Systems (RRS); Summary of feasibility studies and potential standardization topics." Tech. Rep. v1.1.1, October 2009.

[8] R. Prasad and Y. K. Kim. *4G Roadmap and Emerging Communication Technologies*. Universal Personal Communications. Artech House, 2006.

[9] International Telecommunications Union. "All about the Technology.", April 2011. URL `http://www.itu.int/osg/spu/ni/3G/technology/index.html`. [Last Accessed: 27-12-2011].

[10] M. Steer. "Beyond 3G." *Microwave Magazine, IEEE*, vol. 8, no. 1, pp. 76 –82, feb. 2007.

[11] M. L. Roberts, M. A.Temple, R. F. Mills, and R. A. Raines. "Evolution of the air interface of cellular communications systems toward 4G realization." *Communications Surveys Tutorials, IEEE*, vol. 8, no. 1, pp. 2 –23, 2006.

[12] M. Ergen. *Mobile Broadband: Including WiMAX and LTE*. Springer, 2009.

[13] R. Horak. *Telecommunications and Data Communications Handbook*. John Wiley & Sons, Ltd., 2007.

[14] A. F. Molisch. *Wireless Communications*. John Wiley & Sons, Ltd., 2005.

[15] S. A. Ahson and M. Ilyas, editors. *Get Certified: A Guide to Wireless Communication Engineering Technologies*. CRC Press, 2009.

[16] T. Rappaport. *Wireless Communications: Principles and Practice*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2nd ed., 2001.

[17] E. Dahlman, S. Parkvall, J. Sköld, and P. Beming. *3G Evolution: HSPA and LTE for Mobile Broadband*. Elsevier, 2nd ed., 2008.

[18] GSM Association. "Brief History of GSM & the GSMA." URL `http://www.gsma.com/history/`. [Last accessed: 6 February 2012].

[19] M. Sauter. *From GSM to LTE: An introduction to Mobile Networks and Mobile Broadband*. John Wiley & Sons, Ltd., 2011.

[20] Third Generation Partnership Project (3GPP). URL `http://www.3gpp.org/`. [Last Accessed: 6 February 2012].

[21] Third Generation Partnership Project 2 (3GPP2). URL `http://www.3gpp2.org/`. [Last Accessed: 6 February 2012].

[22] J. G. Andrews, A. Ghosh, and R. Muhamed. *Fundamentals of WiMAX: Understanding Broadband Wireless Networking*. Prentice Hall, 2007.

[23] R. van den Bergh and H. Hanrahan. "Evaluation of 3GPP LTE and IEEE 802.16 as Candidate IMT-Advanced Systems." In *Southern Africa Telecommunication Networks and Applications Conference (SATNAC)*. 2008.

[24] L. Nuaymi. *WiMAX: Technology for Broadband Wireless Access*. John Wiley & Sons, Ltd., 2007.

[25] S. Acharya and V. Timofeev. "ITU defines the future of mobile communications: ITU Radiocommunication Assembly approves new developments for its 3G standards.", October 2007. URL `http://www.itu.int/newsroom/press_releases/2007/30.html`. [Last Accessed 6 February 2012].

[26] G. Lawton. "What lies ahead for cellular technology?" *Computer*, vol. 38, no. 6, pp. 14 – 17, jun 2005.

[27] A. R. Prasad. "The Future Re-visited." *Springer Wireless Personal Communications*, vol. 37, pp. 187–211, May 2006.

[28] W. Webb. *Wireless Communications: The Future*. John Wiley & Sons, Ltd., 2007.

[29] International Telecommunications Union. "ITU-R M.1645: Framework and overall objectives of the future development of IMT-2000 and systems beyond IMT-2000." Tech. rep., International Telecommunications Union, 2003.

[30] F. Khan. *LTE for 4G Mobile Broadband: Air Interface Technologies and Performance*. Cambridge University Press, 2009.

[31] P. Lescuyer and T. Lucidarme. *Evolved Packet System (EPS): The LTE and SAE Evolution of 3G UMTS*. John Wiley & Sons, Ltd., 2008.

[32] H. Holma and A. Toskala. *LTE for UMTS: Evolution to LTE-Advanced*. John Wiley & Sons, Ltd., 2nd ed., 2011.

[33] International Telecommunications Union. "What really is a Third Generation (3G) Mobile Technology." Tech. rep., International Telecommunications Union, July 2008. URL `http://www.itu.int/ITU-D/imt-2000/DocumentsIMT2000/What_really_3G.pdf`. [Last accessed: 6 February 2012].

[34] International Telecommunications Union. "ITU-R M.2133: Requirements, evaluation criteria and submission templates for the development of IMT-Advanced." Tech. rep., December 2008.

[35] International Telecommunications Union. "ITU-R M.2134: Requirements related to technical performance for IMT-Advanced radio interface(s)." Tech. rep., December 2008.

[36] S. Acharya and G. Petrin. "IMT-Advanced standards announced for next-generation mobile technology: Specifications for ITU-R recommendation agreed by Radio Assembly.", January 2012. URL `http://www.itu.int/net/pressoffice/press_releases/2012/02.aspx`. [Last Accessed 1 February 2012].

[37] S. Acharya and G. Petrin. "ITU World Radiocommunication Seminar highlights future communication technologies : Focus on international regulations for spectrum management and satellite orbits.", December 2010. URL `http://www.`

itu.int/net/pressoffice/press_releases/2010/48.aspx. [Last Accessed 1 February 2012].

[38] S. Sesia, I. Toufik, and M. Baker, editors. *LTE The UMTS Long Term Evolution: From Theory to Practice*. John Wiley & Sons, Ltd., 2009.

[39] K. Krechmer. "Open standards: A call for change." *Communications Magazine, IEEE*, vol. 47, no. 5, pp. 88 –94, may 2009.

[40] ETSI Technical Committee for RRSs. "TR 102-681: Reconfigurable Radio Systems (RRS); Radio Base Station (RBS) Software Defined Radio (SDR) status, implementations and costs aspects, including future posibilities." Tech. Rep. v1.1.1, June 2009.

[41] International Telecommunications Union. "ITU-R SM.2152: Definitions of Software Defined Radio (SDR) and Cognitive Radio System (CRS)." Tech. rep., International Telecommunications Union, September 2009.

[42] ETSI Technical Committee for RRSs. "TR 102-680: Reconfigurable Radio Systems (RRS); SDR Reference Architecture for Mobile Device." Tech. Rep. v1.1.1, March 2009.

[43] ETSI Technical Committee for RRSs. "TR 102-682: Reconfigurable Radio Systems (RRS); Functional Architecture (FA) for the Management and Control of Reconfigurable Radio Systems." Tech. Rep. v1.1.1, July 2009.

[44] ETSI Technical Committee for RRSs. "TR 102-683: Reconfigurable Radio Systems (RRS); Cognitive Pilot Channel (CPC)." Tech. Rep. v1.1.1, September 2009.

[45] "What is Software Defined Radio." Tech. rep., Wireless Innovation Forum, 2008. URL http://www.wirelessinnovation.org/assets/documents/SoftwareDefinedRadio.pdf. "[last access 13-12-2011]".

[46] E. Grayver. "Standardization efforts for software-defined radio." In *Aerospace Conference, 2010 IEEE*, pp. 1 –8. march 2010.

[47] W. Tuttlebee, editor. *Software Defined Radio: Enabling Technologies*. John Wiley & Sons, Ltd., 2002.

[48] "What are Cognitive Radio and Dynamic Spectrum Access." Tech. rep., Wireless Innovation Forum, 2009. URL http://www.wirelessinnovation.org/assets/documents/Introduction%20to%20CR%20and%20DSA.pdf. "[last access 22-12-2011]".

[49] F. K. Jondral. "Software-Defined Radio - Basics and Evolution to Cognitive Radio." *EURASIP Journal on Wireless Communications and Networking*, vol. 3, pp. 275–283, 2005.

[50] J. Bard and V. J. Kovarik. *Software Defined Radio: The Software Communications Architecture*. John Wiley & Sons, Ltd., 2007.

[51] J. I. Mitola. "Software radios-survey, critical evaluation and future directions." In *Telesystems Conference, 1992. NTC-92., National*, pp. 13/15 –13/23. may 1992.

[52] Joint Program Executive Office (JPEO), Joint Tactical Radio System (JTRS), Space and Naval Warfare Systems Center, U.S. Department of Defense. "Software Communications Architecture Specification 2.2.2.", May 2006. URL `http://groups.winnforum.org/d/do/3766`. [Last accessed: 1 February 2012].

[53] J. Mitola. *Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio*. Ph.D. thesis, Royal Institute of Technology, Sweden, 2000. URL `http://web.it.kth.se/~maguire/jmitola/Mitola_Dissertation8_Integrated.pdf`.

[54] J. Mitola. "Cognitive radio architecture evolution: annals of telecommunications." *Annals of Telecommunications*, vol. 64, pp. 419–441, 2009. URL `http://dx.doi.org/10.1007/s12243-009-0101-6`. 10.1007/s12243-009-0101-6.

[55] D. Klaus, editor. *Technologies for the Wireless Future*, vol. 3. John Wiley & Sons, Ltd., 2008.

[56] IEEE Standards Coordinating Committee 41 on Dynamic Spectrum Access Networks. "IEEE Std 1900.4: IEEE Standard for Architectural Building Blocks Enabling Network-Device Distributed Decision Making for Optimized Radio Resource Usage in Heterogeneous Wireless Access Networks.", 2009.

[57] EU FP7 ICT Project: End-to-End Efficiency (E3). "E^3 Deliverable D2.3: Architecture, Information Model and Reference Points, Assessment Framework, Platform Independent Programmable Interfaces." Tech. rep., 2009.

[58] Z. Boufidis, R. Falk, N. Alonistioti, E. Mohyeldin, N. Olaziregi, and B. Souville. "Network support modeling, architecture, and security considerations for composite reconfigurable environments." *Wireless Communications, IEEE*, vol. 13, no. 3, pp. 34 – 44, june 2006.

[59] V. Stavroulaki, S. Buljore, P. Roux, and E. Melin. "Equipment management issues in B3G, end-to-end reconfigurable systems." *Wireless Communications, IEEE*, vol. 13, no. 3, pp. 24 – 32, june 2006.

[60] M. Mueck, A. Piipponen, K. Kallioja andrvi, G. Dimitrakopoulos, K. Tsagkaris, P. Demestichas, F. Casadevall, J. Pe andrez Romero, O. Sallent, G. Baldini, S. Filin, H. Harada, M. Debbah, T. Haustein, J. Gebert, B. Deschamps, P. Bender, M. Street, S. Kandeepan, J. Lota, and A. Hayar. "ETSI reconfigurable radio systems: status and future directions on software defined radio and cognitive radio standards." *Communications Magazine, IEEE*, vol. 48, no. 9, pp. 78 –86, sept. 2010.

[61] M. Mueck, K. Kalliojarvi, A. Piipponen, J. Gebert, G. Baldini, P. Muller, L. Guang, K. Tsagkaris, P. Demestichas, S. Filin, H. Harada, S. Kandeepan, A. Biswas, M. Stamatelatos, N. Alonistioti, A. Hayar, M. Debbah, and T. Haustein. "ETSI RRS - The standardization path to next generation cognitive radio systems." In *Personal, Indoor and Mobile Radio Communications Workshops (PIMRC Workshops), 2010 IEEE 21st International Symposium on*, pp. 9 –14. sept. 2010.

[62] K. Richardson, C. Jimenez, and D. Stephens. "Evolution of the software communication architecture standard." In *Military Communications Conference, 2009. MILCOM 2009. IEEE*, pp. 1 –8. oct. 2009.

[63] C. Aguayo Gonzalez, C. Dietrich, and J. Reed. "Understanding the software communications architecture." *Communications Magazine, IEEE*, vol. 47, no. 9, pp. 50 –57, september 2009.

[64] PrismTech. "Spectra SDR." URL `http://www.prismtech.com/spectra`. [Last accessed: 8 February 2012].

[65] Object Management Group. "PIM and PSM for Software Radio Components Specification (version 1.0).", March 2007.

[66] D. Paniscotti and J. Bickle. "Evolution and Standardization of the Software Communications Architecture." In *6th SDR Technical Conference and Product Exposition, SDR Forum*. 2006.

[67] A. Kaloxylos, T. Rosowski, K. Tsagkaris, J. Gebert, E. Bogenfeld, P. Magdalinos, A. Galani, and K. Nolte. "The E3 architecture for future cognitive mobile networks." In *Personal, Indoor and Mobile Radio Communications, 2009 IEEE 20th International Symposium on*, pp. 1601 –1605. sept. 2009.

[68] G. Dimitrakopoulos, P. Demestichas, A. Saatsakis, K. Tsagkaris, A. Galani, J. Gebert, and K. Nolte. "Functional Architecture: Management and Control of Reconfigurable Radio Systems." *Vehicular Technology Magazine, IEEE*, vol. 4, no. 1, pp. 42 –48, march 2009.

[69] S. Buljore, H. Harada, S. Filin, P. Houze, K. Tsagkaris, O. Holland, K. Nolte, T. Farnham, and V. Ivanov. "Architecture and enablers for optimized radio resource usage in heterogeneous wireless access networks: The IEEE 1900.4 Working Group." *Communications Magazine, IEEE*, vol. 47, no. 1, pp. 122 –129, january 2009.

[70] S. Filin, H. Harada, H. Murakami, K. Ishizu, and G. Miyamoto. "IEEE 1900.4 WG on architecture and enablers for optimized radio - spectrum resource usage." In *Ultra Modern Telecommunications Workshops, 2009. ICUMT '09. International Conference on*, pp. 1 –8. oct. 2009.

[71] Open Base Architecture Initiative. "BTS System Reference Document.", 2006.

[72] Common Public Radio Interface (CPRI). "Interface Specification v 4.2.", September 2010.

[73] International Telecommunications Union. "X.200: Information technology - Open Systems Interconnection - Basic Reference Model: The basic model." URL `http://www.itu.int/rec/T-REC-X.200-199407-I/en`.

[74] M. Siebert. "Design of a generic protocol stack for an adaptive terminal." In *Proceedings of the Karlsxuhe Workshop on Software Radios*. Karlsruhe, Germany, 2000.

[75] L. Berlemann, A. Cassaigne, and B. Walke. "Generic protocol functions for design and simulative performance evaluation of the link-layer for Reconfigurable Wireless Systems." In *International Symposium on Wireless Personal Multimedia Communications*. 2004.

[76] L. Berlemann, A. Cassaigne, R. Pabst, and B. Walke. "Modular link layer functions of a generic protocol stack for future wireless networks." In *4th SDR Technical Conference and Product Exposition, SDR Forum*. 2004.

[77] V. Gazis, E. Patouni, N. Alonistioti, and L. Merakos. "A survey of dynamically adaptable protocol stacks." *Communications Surveys Tutorials, IEEE*, vol. 12, no. 1, pp. 3 –23, quarter 2010.

[78] L. Berlemann, R. Pabst, M. Schinnenburg, and B. Walke. "A Flexible Protocol Stack for Multi-Mode Convergence in a Relay-based Radio Network

Architecture." In *16th IEEE International Sympossium on Personal, Indoor and Mobile Radio Communications*. 2005.

[79] V. Gazis, N. Alonistioti, and L. Merakos. "A generic model for reconfigurable protocol stacks in beyond 3G." *Wireless Communications, IEEE*, vol. 13, no. 3, pp. 70 – 78, june 2006.

[80] L. Berlemann, R. Pabst, and B. Walke. "Multimode Communication Protocols Enabling Reconfigurable Radios." *EURASIP Journal on Wireless Communications and Networking*, vol. 3, pp. 390–400, 2005.

[81] N. Alonistioti, E. Patouni, and V. Gazis. "Generic Architecture and Mechanisms for Protocol Reconfiguration." *Springer Mobile Networks and Applications*, vol. 11, pp. 917–934, 2006.

[82] GNU Radio. URL `http://gnuradio.org/redmine/projects/gnuradio/wiki`. [Last accessed: 7 February 2012].

[83] The Network Simulator - ns-2. URL `http://www.isi.edu/nsnam/ns/`. [Last accessed: 8 February 2012].

[84] Network Simulator - NS-3. URL `http://www.nsnam.org/`. [Last accessed: 8 February 2012].

[85] Mathworks. "Simulink - Simulation and Model-Based Design." URL `http://www.mathworks.com/products/simulink/`. [Last accessed: 8 February 2012].

[86] L. Berlemann, R. Pabst, and B. Walke. "Efficient multimode protocol architecture for complementary radio interfaces in relay-based 4G networks." *Wireless Communications, IEEE*, vol. 13, no. 3, pp. 15 – 23, june 2006.

[87] M. Siebert and B. Walke. "Design of a generic and adaptive protocol software (DGAPS)." In *Conference on Third Generation Wireless and Beyond*. 2001.

[88] T. Scholer and C. Muller-Schloer. "Design, implementation and validation of a generic and reconfigurable protocol stack framework for mobile terminals." In *Distributed Computing Systems Workshops, 2004. Proceedings. 24th International Conference on*, pp. 362 – 367. march 2004.

[89] B. Walke, L. Berlemann, and D. Schultz. "Architecture Proposal for the WINNER Radio Access Network and Protocol." In *World Wireless Research Forum Meeting 11*. 2004.

[90] V. Stravroulaki, P. Demestichas, L. Berlemann, T. Dodgson, and J. Brakensiek. "Element Management, Flexible Air Interfaces, SDR." Tech. rep., Wireless World Research Forum, 2004.

[91] M. Schinnenburg, R. Pabst, K. Klagges, and B. Walke. "A Software Architecture for Modular Implementation of Adaptive Protocol Stacks." In *MMBnet Workshop*. 2007.

[92] R. van den Bergh and H. Hanrahan. "A modular approach for the development of a reconfigurable access network architecture." In *Southern Africa Telecommunication Networks and Applications Conference (SATNAC)*. 2009.

[93] R. van den Bergh and H. Hanrahan. "Managing the Converged Reconfigurable Radio System Architecture." In *Southern Africa Telecommunication Networks and Applications Conference (SATNAC)*. 2010.

[94] M. C. Jeruchim, P. Balaban, and S. K. Shanmugan. *Simulation of Communication Systems: Modeling, Methodology, and Techniques*. Kluwer Academic / Plenum Publishers, 2nd ed., 2000.

[95] N. Swamy, M and K.-L. Du. *Wireless Communication Systems: From RF Subsystems to G Enabling Technologies*. Cambridge University Press, 2010.

[96] "IEEE Standard for Local and Metropolitan Area Networks Part 16: Air Interface for Fixed Broadband Wireless Access Systems." *IEEE Std 802.16-2009 (Revision of IEEE Std 802.16-2004)*, 2009.

[97] 3rd Generation Partnership Project. "TS 36.300: Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2 (Release 9).", December 2010.

[98] D. Tse and P. Viswanath. *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.

[99] Y. Zhang and H.-H. Chen, editors. *Mobile WiMAX: Toward Broadband Wireless Metropolitan Area Networks*. Auerback Publications, Talor & Francis Group, 2008.

[100] IEEE 802.16 Working Group. "Five Criteria Statement for P802.16m PAR Proposal." Tech. rep., 2006. URL `http://ieee802.org/16/docs/06/80216-06_055r3.pdf`. [Last accessed: 7 February 2012].

[101] "IEEE Standard for Local and metropolitan area networks Part 16: Air Interface for Broadband Wireless Access Systems Amendment 3: Advanced Air

Interface." *IEEE Std 802.16m-2011(Amendment to IEEE Std 802.16-2009)*, pp. 1 –1112, 5 2011.

[102] WiMAX Forum. "WiMAX Network Architecture Specifications By Release." URL `http://www.wimaxforum.org/resources/documents/technical/release`. [Last accessed: 7 February 2012].

[103] WiMAX Forum. "WiMAX Forum Certification Program." URL `http://www.wimaxforum.org/certification/certification-overview`. [Last accessed: 7 February 2012].

[104] Third Generation Partnership Project (3GPP). "Release 10." URL `http://www.3gpp.org/Release-10`. [Lasr acccessed: 7 February 2012].

[105] 3rd Generation Partnership Project. "TS 36.323: Evolved Universal Terrestrial Radio Access (E-UTRA); Packet Data Convergence Protocol (PDCP); Protocol specification (Release 9).", December 2009.

[106] 3rd Generation Partnership Project. "TS 36.322: Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Link Control (RLC); Protocol specification (Release 9).", September 2009.

[107] 3rd Generation Partnership Project. "TS 36.321: Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC); Protocol specification (Release 9).", June 2010.

[108] 3rd Generation Partnership Project. "TS 36.201: Evolved Universal Terrestrial Radio Access (E-UTRA); LTE physical layer; General description (Release 9).", March 2010.

[109] 3rd Generation Partnership Project. "TS 36.311: Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification (Release 9).", December 2010.

[110] 3rd Generation Partnership Project. "TS 36.212: Evolved Universal Terrestrial Radio Access (E-UTRA); Physical Channels and Modulation (Release 9).", March 2010.

[111] 3rd Generation Partnership Project. "TS 36.212: Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding (Release 9).", September 2010.

[112] H. Schildt. *Java: The Complete Reference*. McGraw-Hill, 7th ed., 2007.

[113] Oracle Corporation. "The Java Tutorials: The Reflection API.", December 2011. URL http://docs.oracle.com/javase/tutorial/reflect/index.html. [Last accessed: 7 February 2012].

[114] Telecom Italia. "Java Agent Development Framework (JADE)." URL http://jade.tilab.com/. [Last accessed: 7 February 2012].

[115] F. Bellifemine, G. Caire, and G. Dominic. *Developing Multi-Agent Systems with JADE*. John Wiley & Sons, Ltd., 2007.

[116] MathWorks. "Communications System Toolbox." URL http://www.mathworks.com/products/communications/. [Last accessed: 7 February 2012].

[117] MathWorks. "Signal Processing Toolbox." URL http://www.mathworks.com/products/signal/. [Last accessed: 7 February 2012].

[118] MathWorks. "MATLAB Builder JA." URL http://www.mathworks.com/products/javabuilder/. [Last accessed: 7 February 2012].

[119] J. Kaplan. "MATLABControl: A Java API to interact with MATLAB v4.", July 2011. URL http://code.google.com/p/matlabcontrol/. [Last accessed: 7 February 2012].

[120] Ettus Research. "About Ettus Research." URL http://www.ettus.com/site/about. [Last accessed: 7 February 2012].

[121] Ettus Research. "The USRP Hardware Driver (UHD)." URL http://ettus-apps.sourcerepo.com/redmine/ettus/projects/uhd/wiki. [Last accessed: 7 February 2012].

[122] Ettus Research. "RF Daughterboards: XCVR2450." URL https://www.ettus.com/product/details/XCVR2450. [Last accessed: 7 February 2012].

[123] World Wide Web Consortium (W3C). "Extensible Markup Language (XML)." URL http://www.w3.org/XML/. [Last accessed: 7 February 2012].

[124] S. Holzner. *XML: A Beginner's Guide*. McGraw-Hill, 2009.

[125] W. Stallings. *Data & Computer Communications*. Prentice Hall, 6th ed., 2000.

[126] I. A. Glover and P. M. Grant. *Digital Communications*. Prentice Hall, 2nd ed., 2004.

[127] J. van de Beek, M. Sandell, and P. Borjesson. "ML estimation of time and frequency offset in OFDM systems." *Signal Processing, IEEE Transactions on*, vol. 45, no. 7, pp. 1800 –1805, jul 1997.

Appendix A

Guide to the CD

The attached CD contains the CRRSA implementation, which includes all policy files, MATLAB functions, the source code for the SDR API C++ wrapper, all Java source code, and the compiled binaries to run the simulation. The directory structure of the CD is shown in Figure A.1.

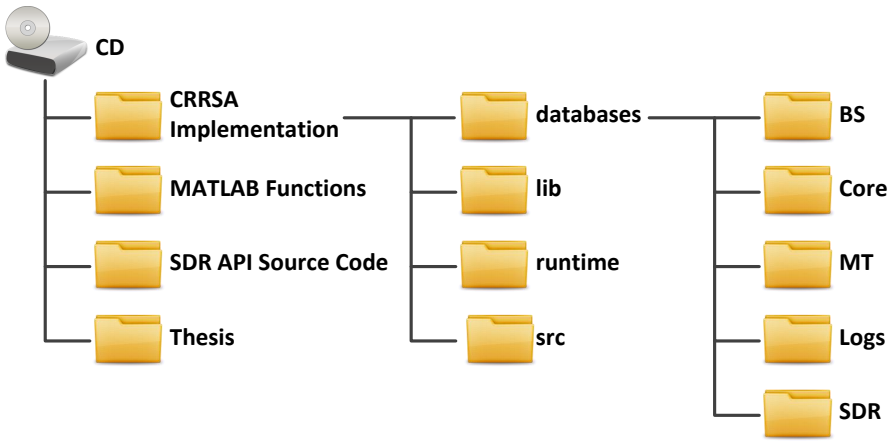


Figure A.1: Directory structure of attached CD

The contents of the CD are organized as follows:

- **CRRSA Implementation**: contains the Java source code, all policy files, and compiled binaries of the implementation.
- **MATLAB Functions**: contains the MATLAB functions which implement the signal processing techniques used in Reconfigurable Components. These MATLAB functions were written using MATLAB 2011a (64-bit) together with

the Communications System Toolbox and the Signal Processing Toolbox. To run the CRRSA implementation, MATLAB 2011a (64-bit) must be installed on the target PC.

- **SDR API Source Code**: contains the C++ and Java source code files for the SDR API C++ Wrapper. A 64-bit binary of the UHD library from Ettus is required to compile these files. Although Ettus does not provide a 64-bit binary version of the UHD library on their website, they do provide the instructions to compile the UHD from source code [1].

- **Thesis**: contains an electronic copy of this document and requires Adobe Acrobat Reader to view the file.

A series of Windows batch files have been created to assist in running the CRRSA implementation. These batch files contain the simulation parameters used to setup a No Wireless Channel Scenario, a MATLAB Simulated Wireless Channel Scenario, and a Real Wireless Channel using two USRP SDR hardware platforms. Another batch file which presents a GUI is also provided, in the event that the user wishes to run a simulation with alternate parameters. All batch files must be run with administrator privileges, and all directories must be writable.

The CRRSA implementation was developed using a Windows 7 64-bit operating system with 6GB of RAM. Although the majority of the simulation is written in Java and will therefore run on any computer architecture, the SDR API C++ Wrapper library is not platform independent and will only work on a Windows 7 64-bit operating system. In addition, to run any SDR based simulation two USRP SDR hardware platforms must be provided.

The contents of the CRRSA implementation directory are organized as follows:

- **databases**: contains the Policy, and Reconfigurable Component and Algorithm databases for the Base Station, Mobile Terminal, and Core Network. These databases include the XML Network Equipment and RAT policy files, and the Reconfigurable Component and Algorithm Java files. Separate directories contain the log files generated during a simulation, and the SDR API C++ Wrapper library.

[1] Build instructions for the UHD library in Windows: `http://files.ettus.com/uhd_docs/manual/html/build.html` [Last accessed: 9 February 2012]

- **lib**: contains all third-party software libraries used in the CRRSA implementation.
- **runtime**: used during a simulation to store the Java Class files created when Reconfigurable Component and Algorithm Java files are compiled.
- **src**: contains the Java source code for the CRRSA implementation.

Appendix B

List of Reconfigurable Components

In order to demonstrate that the CRRSA is capable of implementing and reconfiguring RATs at runtime, two RAT models were developed based on the functionality and signal processing techniques contained in the IEEE 802.16 and 3GPP LTE RAT standards. These RAT models are constructed using Reconfigurable Components which contain the fundamental elements of each of the chosen RATs.

Tables B.1, B.2, B.3, B.4, and B.5 contain a list of all the generic and RAT specific Reconfigurable Components used to construct RATs in the CRRSA implementation, and specify each Component's input and output parameters, implementation language and/or method, and the context information submitted to the RMM.

Tables B.6 and B.7 contain a list of all the Abstracted Control Reconfigurable Components which provide statically defined and random parameters to the generic and RAT specific Reconfigurable Components in each RAT, and specify each Component's output parameters and values.

Table B.1: Generic and RAT specific Reconfigurable Components used to construct RATs in the CRRSA implementation

Reconfigurable Component	Implementation Details	Input Parameters			Output Parameters		Implementation Language / Method	Context Information				
		Name	Independent / Dependent	Bearer/Control	Name	Bearer/Control		No. Rx Packets	No. Rx Bits	No. Dropped Packets	Average Processing Time	RSSI
Append CRC	Calculates and appends the CRC value of the input packet	Packet	Independent	Bearer	Packet	Bearer	MATLAB Builder JA	No	No	No	Yes	No
		CRC Generator Polynomial	Independent	Control								
Channel Quality Measurement	Requests the RSSI measurement from the WMUX and reports it to the monitoring module	N/A		N/A	N/A	N/A	Java	No	No	No	Yes	Yes
Convolutional Coding with Tail Biting	Encodes the bits in the input packet using Convolutional Coding with Tail Biting	Packet	Independent	Bearer	Packet	Bearer	MATLAB Control API	No	No	No	Yes	No
		Constraint Length	Independent	Control								
		Generator Polynomial	Independent	Control								
		Puncture Pattern	Independent	Control								
		Code Rate	Independent	Control								
Convolutional Decoding with Tail Biting	Decodes the bits in the input packet using Convolutional Decoding with Tail Biting	Packet	Independent	Bearer	Packet	Bearer	MATLAB Control API	No	No	No	Yes	No
		Constraint Length	Independent	Control								
		Generator Polynomial	Independent	Control								
		Puncture Pattern	Independent	Control								
		Code Rate	Independent	Control								
Convolutional Turbo Coding	Encodes the bits in the input packet using Convolutional Turbo Coding.	Packet	Independent	Bearer	Packet	Bearer	MATLAB Control API	No	No	No	Yes	No
		Convolutional Encoder 1 Constraint Length	Independent	Control								
		Convolutional Encoder 1 Generator Polynomial	Independent	Control								
		Convolutional Encoder 1 Feedback Connections	Independent	Control								
		Convolutional Encoder 2 Constraint Length	Independent	Control								
		Convolutional Encoder 2 Generator Polynomial	Independent	Control								
		Convolutional Encoder 2 Feedback Connections	Independent	Control								
		Interleaver Parameter 1	Independent	Control								
		Interleaver Parameter 2	Independent	Control								

Table B.2: Generic and RAT specific Reconfigurable Components used to construct RATs in the CRRSA implementation

Reconfigurable Component	Implementation Details	Input Parameters			Output Parameters		Implementation Language / Method	Context Information				
		Name	Independent / Dependent	Bearer/Control	Name	Bearer/Control		No. Rx Packets	No. Rx Bits	No. Dropped Packets	Average Processing Time	RSSI
Convolutional Turbo Decoding	Decodes the bits in the input packet using Convolutional Turbo Decoding.	Packet	Independent	Bearer	Packet	Bearer	MATLAB Control API	No	No	No	Yes	No
		Convolutional Encoder 1 Constraint Length	Independent	Control								
		Convolutional Encoder 1 Generator Polynomial	Independent	Control								
		Convolutional Encoder 1 Feedback Connections	Independent	Control								
		Convolutional Encoder 2 Constraint Length	Independent	Control								
		Convolutional Encoder 2 Generator Polynomial	Independent	Control								
		Convolutional Encoder 2 Feedback Connections	Independent	Control								
		Interleaver Parameter 1	Independent	Control								
		Interleaver Parameter 2	Independent	Control								
		No. Decoding Iterations	Independent	Control								
Detect Errors and Remove CRC	Removes the CRC value appended to the input packet, calculates the CRC value for the remaining packet, and compares the two CRC values. If the values match, the packet has not been corrupted.	Packet	Independent	Bearer	Packet	Bearer	MATLAB Builder JA	Yes	Yes	Yes	Yes	No
		CRC Generator Polynomial	Independent	Control								
Gold Randomiser	Generates a Gold code which is XOR'd with the bits in the input packet. Used by LTE to perform scrambling.	Packet	Independent	Bearer	Packet	Bearer	MATLAB Builder JA	No	No	No	Yes	No
		Generator Polynomial 1	Independent	Control								
		Initial Condition 1	Independent	Control								
		Generator Polynomial 2	Independent	Control								
		Initial Condition 2	Independent	Control								
		Code delay	Independent	Control								
Least Squares Channel Estimation	Estimates the Channel Transfer Function from transmitted and received Pilot symbols	Packet	Independent	Bearer	Packet	Bearer	MATLAB Builder JA	No	No	No	Yes	No
		Subcarrier Permutation	Independent	Control	Estimated Channel Transfer Function	Control						
		Transmitted Pilot Symbols	Independent	Control	Estimated Channel Transfer Function for Data Subcarriers only	Control						
		Received Pilot Symbols	Independent	Control								

Table B.3: Generic and RAT specific Reconfigurable Components used to construct RATs in the CRRSA implementation

Reconfigurable Component	Implementation Details	Input Parameters			Output Parameters		Implementation Language / Method	Context Information				
		Name	Independent / Dependent	Bearer/Control	Name	Bearer/Control		No. Rx Packets	No. Rx Bits	No. Dropped Packets	Average Processing Time	RSSI
LTE Append MAC Header	Generates and appends the LTE MAC header using the supplied parameters	Packet	Independent	Bearer	Packet	Bearer	Java	No	No	No	Yes	No
		LTE Logical Channel ID	Independent	Control								
LTE Remove MAC Header	Removes the LTE MAC header	Packet	Independent	Bearer	Packet	Bearer	Java	No	No	No	Yes	No
M-QAM Demodulation	Demodulates M-QAM Symbols into bits	Active Reconfigurable Component	Independent	Control	Packet	Bearer	MATLAB Builder JA	No	No	No	Yes	No
		Packet	Dependent	Bearer								
		Digital Alphabet Size	Dependent	Control								
		Modulation Input Type	Dependent	Control								
M-QAM Modulation	Modulates bits in the input packet into M-QAM Symbols	Active Reconfigurable Component	Independent	Control	Packet	Bearer	MATLAB Builder JA	No	No	No	Yes	No
		Packet	Dependent	Bearer								
		Digital Alphabet Size	Dependent	Control								
		Modulation Input Type	Dependent	Control								
		Scaling Factor	Dependent	Control								
OFDM Demodulation	Removes the cyclic prefix and performs the FFT operation on the received packet resulting in the demodulated subcarriers	Packet	Independent	Bearer	Subcarriers	Bearer	MATLAB Builder JA	No	No	No	Yes	No
		Sample Rate	Independent	Control								
		Subcarrier Spacing	Independent	Control								
		Cyclic Prefix Ratio	Independent	Control								
OFDM Modulation	Performs the IFFT operation on the received subcarriers and prepends the Cyclic Prefix	Subcarriers	Independent	Bearer	Packet	Bearer	MATLAB Builder JA	No	No	No	Yes	No
		Sample Rate	Independent	Control								
		Subcarrier Spacing	Independent	Control								
		Cyclic Prefix Ratio	Independent	Control								
Packet Assembler	Buffers a specified number of consecutive packets, contenates them, and outputs the combined frame	Packet	Independent	Bearer	Packet	Bearer	Java	No	No	No	Yes	No
		Number of Input Packets to each Output Packet	Independent	Control								
Packet Dissassembler	Splits the supplied packet into a specified number of smaller packets, and then buffers these packets in its output queue	Packet	Independent	Bearer	Packet	Bearer	Java	No	No	No	Yes	No
		Number of Output Packets to each Input Packet	Independent	Control								
QPSK Demodulation	Demodulates QPSK Symbols into bits	Active Reconfigurable Component	Independent	Control	Packet	Bearer	MATLAB Control API	No	No	No	Yes	No
		Packet	Dependent	Bearer								
		Modulation Input Type	Dependent	Control								

Table B.4: Generic and RAT specific Reconfigurable Components used to construct RATs in the CRRSA implementation

Reconfigurable Component	Implementation Details	Input Parameters			Output Parameters		Implementation Language / Method	Context Information				
		Name	Independent / Dependent	Bearer/Control	Name	Bearer/Control		No. Rx Packets	No. Rx Bits	No. Dropped Packets	Average Processing Time	RSSI
QPSK Modulation	Modulates bits in the input packet into QPSK Symbols	Active Reconfigurable Component	Independent	Control	Packet	Bearer	MATLAB Builder JA	No	No	No	Yes	No
		Packet	Dependent	Bearer								
		Modulation Input Type	Dependent	Control								
Randomiser	Generates a Pseudo random binary sequence which is XOR'd with the bits in the input packet. Used by 802.16 to perform randomisation.	Packet	Independent	Bearer	Packet	Bearer	MATLAB Control API	No	No	No	Yes	No
		Generator Polynomial	Independent	Control								
		Initial Condition	Independent	Control								
Subcarrier Mapping	Maps the supplied data and pilot symbols onto subcarriers according to the supplied subcarrier permutation pattern	Packet	Independent	Bearer	Subcarriers	Bearer	MATLAB Builder JA	No	No	No	Yes	No
		Pilot Symbols	Independent	Control								
		Subcarrier Permutation	Independent	Control								
Subcarrier Unmapping	Retrieves the data and pilot symbols from the supplied subcarriers using supplied subcarrier permutation pattern	Subcarriers	Independent	Bearer	Packet	Bearer	MATLAB Builder JA	No	No	No	Yes	No
		Subcarrier Permutation	Independent	Control	Pilot Symbols	Control						
Two Channel Demultiplexer	Retrieves only one of its dependent inputs based on its independent control input, and outputs that depended input	MUX Selection Control	Independent	Control	Packet	Bearer	Java	No	No	No	Yes	No
		Packet 1	Dependent	Bearer								
		Packet 2	Dependent	Bearer								
WiMAX Append Generic MAC Header	Generates and appends the 802.16 Generic MAC header using the supplied parameters	Packet	Independent	Bearer	Packet	Bearer	MATLAB Control API	No	No	No	Yes	No
		Is Encryption Used	Independent	Control								
		Is ARQ Used	Independent	Control								
		Is PSH or FSH Extended	Independent	Control								
		Is FSH Used	Independent	Control								
		Is PSH Used	Independent	Control								
		Is FFSH or GMSH Used	Independent	Control								
		Is ESF Used	Independent	Control								
		Is CRC Used	Independent	Control								
		Encryption Key Sequence	Independent	Control								
		Connection ID	Independent	Control								
WiMAX Interleaver	Interleaves the bits in the received packet according to the unique equations specified by the 802.16 standard	Packet	Independent	Bearer	Packet	Bearer	MATLAB Builder JA	No	No	No	Yes	No
		Digital Alphabet Size for Interleaver	Independent	Control								

Table B.5: Generic and RAT specific Reconfigurable Components used to construct RATs in the CRRSA implementation

Reconfigurable Component	Implementation Details	Input Parameters			Output Parameters		Implementation Language / Method	Context Information				
		Name	Independent / Dependent	Bearer/Control	Name	Bearer/Control		No. Rx Packets	No. Rx Bits	No. Dropped Packets	Average Processing Time	RSSI
WiMAX Deinterleaver	Deinterleaves the bits in the received packet according to the unique equations specified by the 802.16 standard	Packet	Independent	Bearer	Packet	Bearer	MATLAB Builder JA	No	No	No	Yes	No
		Digital Alphabet Size for Deinterleaver	Independent	Control								
WiMAX Remove Generic MAC Header	Removes the 802.16 Generic MAC header	Packet	Independent	Bearer	Packet	Bearer	Java	No	No	No	Yes	No
Zero Forcing Equaliser	Equalises the modulated symbols in the received packet using the Zero Forcing Equalisation method and the supplied Channel Transfer Function	Packet	Independent	Bearer	Packet	Bearer	Java	No	No	No	Yes	No
		Transfer Function for all Subcarriers	Independent	Control								
		Transfer Funcion for only Data Subcarriers	Independent	Control								

Table B.6: Abstracted Control Reconfigurable Components used to construct RATs in the CRRSA implementation

Abstracted Control Reconfigurable Component	Parameter Details	Output Parameter	Value
Generate Random Pilots Symbols	Randomly chosen to demonstrate the functionality provided by other Reconfigurable Components	Pilot Symbols	Randomly Generated BPSK Symbols
Generate Random Subcarrier Permutations	Randomly chosen to demonstrate the functionality provided by other Reconfigurable Components	Subcarrier Permutation	Subcarrier Permutation where Pilot Symbols are evenly spaced across all subcarriers, and the positions of all data subcarriers are determined randomly
LTE MAC PDU Control	Specified as part of the Multiplexing specification in Section 5.1.1 of 3GPP TS 36.212 v9.3.0 September 2010	CRC Generator Polynomial	Excessively large / complex value, please see the standard
	Controlled by Scheduler in a full implementation, this value was chosen to ensure that the sizes of the WiMAX and LTE packets conveniently matched	Number of MAC SDUs per PDU	4
	Randomly chosen to demonstrate the functionality provided by other Reconfigurable Components	LTE Logical Channel ID	00001
LTE Adaptive Modulation and Coding	Randomly chosen from one of of three modulation options specified in 3GPP TS 36.211 v9.1.0 March 2010	Modulation and DeMUX Selection	QAM
		Digital Modulation Alphabet Size	16
		Modulation Input Type	Bit Input
	The LTE standard does not specify a scaling factor	Modulation Scaling Factor	1.0
	Specified as part of the Channel Coding specification in 3GPP TS 36.212 v9.3.0 September 2010	Convolutional Encoder 1 Constraint Length	4
		Convolutional Encoder 1 Generator Polynomial	[13 15]
		Convolutional Encoder 1 Feedback Connections	13
		Convolutional Encoder 2 Constraint Length	4
		Convolutional Encoder 2 Generator Polynomial	[13 15]
		Convolutional Encoder 2 Feedback Connections	13
	Calculated based on the size of the Transport Block and the values in Table 5.1.3-3 in 3GPP TS 36.212 v9.3.0 September 2010	Interleaver Parameter 1	45
		Interleaver Parameter 2	354
	Randomly chosen to demonstrate the functionality provided by other Reconfigurable Components	No. Decoding Iterations	12
LTE Physical Frame Control	The number of total subcarriers and the sampling rate were both chosen according to the 3GPP LTE standard. The rest were randomly chosen to demonstrate the functionality provided by other Reconfigurable Components	No OFDM Symbols per Physical Frame	1
		No Pilot Symbols per OFDM Symbol	473
		Pilot Symbol Scaling Factor	4.0
		No DC Subcarriers per OFDM Symbol	1
		No Left Guard Subcarriers per OFDM Symbol	78
		No Right Guard Subcarriers per OFDM Symbol	77
		No Data Subcarriers per OFDM Symbol	1419
		Sampling Rate	5MHz
		Subcarrier Spacing	1kHZ
		Cyclic Prefix Ratio	0.25
LTE Physical Layer Control	Specified as part of the Physical Channels and Modulation specification in Section 7.2 of 3GPP TS 36.211 v9.1.0 March 2010	Gold Randomiser Generator Polynomial 1	Excessively large / complex value, please see the standard
		Gold Randomiser Initial Condition 1	Excessively large / complex value, please see the standard
		Gold Randomiser Generator Polynomial 2	Excessively large / complex value, please see the standard
		Gold Randomiser Initial Condition 2	Excessively large / complex value, please see the standard
		Gold Randomiser Delay	1600

Table B.7: Abstracted Control Reconfigurable Components used to construct RATs in the CRRSA implementation

Abstracted Control Reconfigurable Component	Parameter Details	Output Parameter	Value
WiMAX MAC PDU Control	Specified in Section 6.3.3.5.2 of IEEE 802.16 - 2009	CRC Generator Polynomial	Excessively large / complex value, please see the standard
	Randomly chosen to demonstrate the functionality provided by other Reconfigurable Components	Is Encryption Used	No
		Is ARQ Used	No
		Is PSH or FSH Extended	No
		Is FSH Used	No
		Is PSH Used	No
		Is FFSH or GMSH Used	No
		Is ESF Used	No
		Is CRC Used	Yes
		Encryption Key Sequence	[0 0]
		Connection ID	32
WiMAX Adaptive Modulation and Coding	Randomly chosen from one of of three modulation options specified in IEEE 802.16 - 2009	Modulation and DeMUX Selection	QPSK
		Digital Modulation Alphabet Size	4
		Modulation Input Type	Bit Input
	Specified in Section 8.4.9.4.2 of IEEE 802.16 - 2009	Modulation Scaling Factor	sqrt(0.5)
	Randomly chosen to demonstrate the functionality provided by other Reconfigurable Components	Code Rate	2/3
	Based on the chosen Code Rate and Section 8.3.3.2.1 of IEEE 802.16 - 2009	Convolutional Encoder Constraint Length	7
		Convolutional Encoder Generator Polynomial	[171 133]
		Convolutional Encoder Puncture Pattern	[1 0 1 1]
	Randomly chosen to demonstrate the functionality provided by other Reconfigurable Components	No. Decoding Iterations	3
WiMAX Physical Frame Control	The number of total subcarriers and the sampling rate were both chosen according to the IEEE 802.16 - 2009 standard. The rest were randomly chosen to demonstrate the functionality provided by other Reconfigurable Components	No OFDM Symbols per Physical Frame	3
		No Pilot Symbols per OFDM Symbol	384
		Pilot Symbol Scaling Factor	4.0
		No DC Subcarriers per OFDM Symbol	49
		No Left Guard Subcarriers per OFDM Symbol	232
		No Right Guard Subcarriers per OFDM Symbol	231
		No Data Subcarriers per OFDM Symbol	1200
		Sampling Rate	5MHz
		Subcarrier Spacing	1kHz
		Cyclic Prefix Ratio	0.25
WiMAX Physical Layer Control	Specified in Section 8.3.3.1 of IEEE 802.16 - 2009	Randomiser Generator Polynomial	Excessively large / complex value, please see the standard
		Randomiser Initial Condition	Excessively large / complex value, please see the standard