UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Implementation of the Physical Random Access Channel in TDD-LTE using Software Defined Radio

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Daniel Stephane Tiati Dang

In partial fulfilment

of the requirements for the degree in

Science and Telecommunication Technologies
ENGINEERING

Advisor: Jordi Perez Romero, Jäntti Riku

Barcelona, February 2016

Abstract

In Long Term Evolution (LTE) uplink transmission, a User Equipment (UE) must be time-synchronized before normal data transmission. So that a Physical Random Access Channel (PRACH) becomes a crucial factor for LTE access scheme since it is the base on which the Random Access Channel (RACH) is implemented.

In this project, a PRACH module for TD-LTE base station system using software defined radio has been implemented. The System follows the release 8 of LTE-UMTS specifications. During the development, a scenario with simple uplink transmission between one UE and one eNodeB has be considered. The UE is able to generate one PRACH sequence and its baseband signal at a time. In the other hand, The eNodeB is able to generate the 64 PRACH sequences available per LTE-cell, process the baseband signal received from the UE and detect the PRACH sequence transmitted in approximately 1ms.

Resum

En Long term Evolution, en la transmissió en enllaç ascendent, un equip d'usuari (UE) ha d'estar sincronitzat en temps abans de la transmissió normal de dades. Per conseqüent, un Physical Random Access Channel (PRACH) es torna un factor crucial per a l'esquema d'access LTE ja que es la base sobre la que s' implementa el Random Access Channel (RACH).

En aquest project, s'ha implementat el mòdul PRACH per a una estació base TD-LTE utilitzant ràdio definida per software. El sistema segueix les especificacions del LTE-UMTS, versió 8. Durant el desenvolupament, s'ha considerat un escenari en el que tenim una simple transmissió en enllaç ascendent entre un UE y un eNodeB. UE és capaç de generar només un PRACH sequence i el seu baseband signal cada cop. L'eNodeB per la seva part es capaç de generar els 64 PRACH sequences disponibles per a cada LTE-cell, processar el baseband signal rebut i trobar en aproximadament 1 ms el PRACH sequence transmès.

Resumen

En Long Term Evolution (LTE), en la transmisión en enlace ascendiente, un equipo de usuario (UE) debe estar sincronizado en tiempo antes de la transmisión normal de datos. Por consiguiente, un Physical Random Access Channel (PRACH) se convierte en un factor crucial para el esquema de acceso LTE ya que es la base sobre la que se implementa el Random Access Channel (RACH).

En este proyecto, se ha implementado el módulo PRACH para una estación base TD-LTE utilizando radio definida por software. El sistema sigue las especificaciones del LTE-UMTS, version 8. Durante el desarrollo, se ha considerado un escenario en el que tenemos una simple transmisión en enlace ascendiente entre un UE y un eNodeB. El UE es capaz de generar solo un PRACH sequence y su baseband signal cada vez. El eNodeB por el otro lado es capaz de generar los 64 PRACH sequences disponibles para cada LTE-cell, procesar el baseband signal recibido y encontrar en aproximadamente 1 ms el PRACH sequence transmitido.

Acknowledgements

Working on this project has immensely broaden my skills and knowledge in radio communication. I am deeply grateful for this amazing opportunity provided by Aalto University and Universitat Politécnica de Catalunya.

I would like to express my profound gratitude to my project supervisor, Dr. Jordi Perez Romero, Dr. Jäntti Riku, and more especially to Dr. Kalle Ruttik who made the success of this project possible.

Finally I strongly feel grateful for the support received from my incredible family, friends in Spain and the amazing people I have met in Finland.

Many Thanks,

Muchas grácias,

Daniel Stephane Tiati Dang

Revision history and approval record

Revision	Date	Purpose
0	29/09/2015	Document creation
1	24/11/2015	Document revision
2	21/01/2016	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Daniel Stephane Tiati Dang	dondanidang@gmail.com
Jordi Perez Romero	jorperez@tsc.upc.edu
Jäntti Riku	Riku.jäntti@aalto.fi
Kalle Ruttik	kalle.ruttik@aalto.fi

Written by:		Reviewed and approved by:	
Date	07/01/2016	Date	25/01/2016
Name	Daniel Stephane Tiati Dang	Name	Jordi Perez Romero
Position	Project Author	Position	Project Supervisor

Table of Contents

List of Figures

Figure 30. frame structure type 2 random access configurations

1. Introduction

Wireless communications have been developing rapidly in recent years. Long Term Evolution (LTE) system and its evolutionary techniques are the mainstreams of the mobile communications systems. Under certain circumstances, an LTE UE can be scheduled for uplink transmission only if uplink transmission timing is synchronized. LTE Random Access Channel (RACH) therefore plays a key role as interface between non-synchronized UEs and the orthogonal transmission scheme of the LTE uplink radio access. However as a first step of RACH, Physical Random Access Channel (PRACH) is critical for the whole random access procedure.

1.1. Project Overview and Goals

The project is carried out at the communication and network department (ComNet) of Aalto University by the LTE research team and it is part of their own LTE testing base station they are building.

The purpose of this project is mainly to implement the Physical Random Access Channel (PRACH) of LTE using software defined radio and following the release 8 of TD-LTE specifications.

LTE, a project of 3rd Generation Partnership Project (3GPP), is standardized to comply with the International Mobile Telecommunication (IMT) Advanced 4th generation requirement. LTE provides high data rate, flexible scheduling, and improved Quality of Service (QoS) by using remarkable technology.

LTE has two access modes: Frequency Division Duplexing (FDD-LTE) and Time Division Duplexing (TDD-LTE or TD-LTE). The difference between both is that FDD-LTE uses two channels for the communication between the eNodeB or LTE base station and the UE, one for the uplink (UL) and the other for the downlink (DL) while the TD-LTE only uses one channel for both UL and DL communications. Given the application where this project will be applied, the TD-LTE was chosen.

The Software Defined Radio (SDR), on the other hand, is the new paradigm in wireless communication to implement, by means of software, the functionalities of many hardware components used in communication system. This brings a lot of flexibility, and both TD-LTE and SDR allow an increases in the spectrum efficiency.

The main goal of this project are:

- Implementation of the PRACH module
- Integration of the PRACH module in the TD-LTE system using software defined radio.

1.2. Requirements and Specifications

- The implementation of the PRACH should follow the release 8 of LTE-UMTS of 3GPP.
- PRACH should be implemented using C++ programming.
- PRACH should be implemented following the Oriented Object Programming (OOP).
- PRACH implementation should be memory and time efficient.
- UE and eNodeB should be able to manages PRACH sequence.
- In the PRACH module integration in the rest of the system, the UE should generate and transmit the PRACH sequence while the eNodeB should be able to recover the PRACH sequence transmitted.
- The implementation should guarantee a quick constant response time (approximately 1ms) of the eNodeB to recover the preamble PRACH transmitted by the UE.

1.3. Work Plan and Gantt Diagram

Work Packages

<table>
<tr><td colspan="2">Project: PRACH implementation using SDR</td><td>WP ref: WP1</td></tr>
<tr><td colspan="2">Major constituent: Software</td><td>Sheet n of m</td></tr>
<tr><td colspan="2" rowspan="2">Learning

Read about LTE and learn how it works. Later focus on the RACH process and SDR. For the implementation I have to learn C++ programming.</td><td>Planned start date: 13/09/2015
Planned end date: 24/09/2015</td></tr>
<tr><td>Start event: 15/09/2015
End event: 01/10/2015</td></tr>
<tr><td>Internal task T1: LTE
- Overview about LTE
- Refresh the concept behind LTE like OFDM and SC-FDMA
- Learn how LTE works and all the parts involved in LTE

Internal task T2: RACH and PRACH
- Deep learning of the RACH process

Internal task T3: SDR
- Read and understand SDR
- Learn how gnuradio works</td><td>Deliverables:</td><td>Dates:</td></tr>
</table>

Internal task T4: C++ - Learn C++ programming - Learn doxygen		

Project:PRACH implementation using SDR	WP ref: WP2	
Major constituent: Software, Hardware	Sheet n of m	
Workspace Setup my workspace and download the last version of git project branch.	Planned start date: 28/09/2015 Planned end date: 05/10/2015	
	Start event: 04/10/2015 End event: 09/10/2015	
Internal task T1: Environment setup - Install Ubuntu - Install Octave - Install GNURadio - install the necessary libraries for proper operation of the software. Internal task T2: Clone git project branch from the server to my workspace	Deliverables:	Dates:

Project: PRACH implementation using SDR	WP ref: WP3	
Major constituent: Software	Sheet n of m	
Testing and improvement of existing PRACH code Get familiar with the existing code by testing. Also test the new code that I will develop.	Planned start date: 11/10/2015 Planned end date: 30/10/2015	
	Start event: End event:	
Internal task T1: Testing existing code Internal task T2: Testing the code improvement	Deliverables:	Dates:

Project:RACH implementation using SDR	WP ref: WP4
Major constituent: Software	Sheet n of m

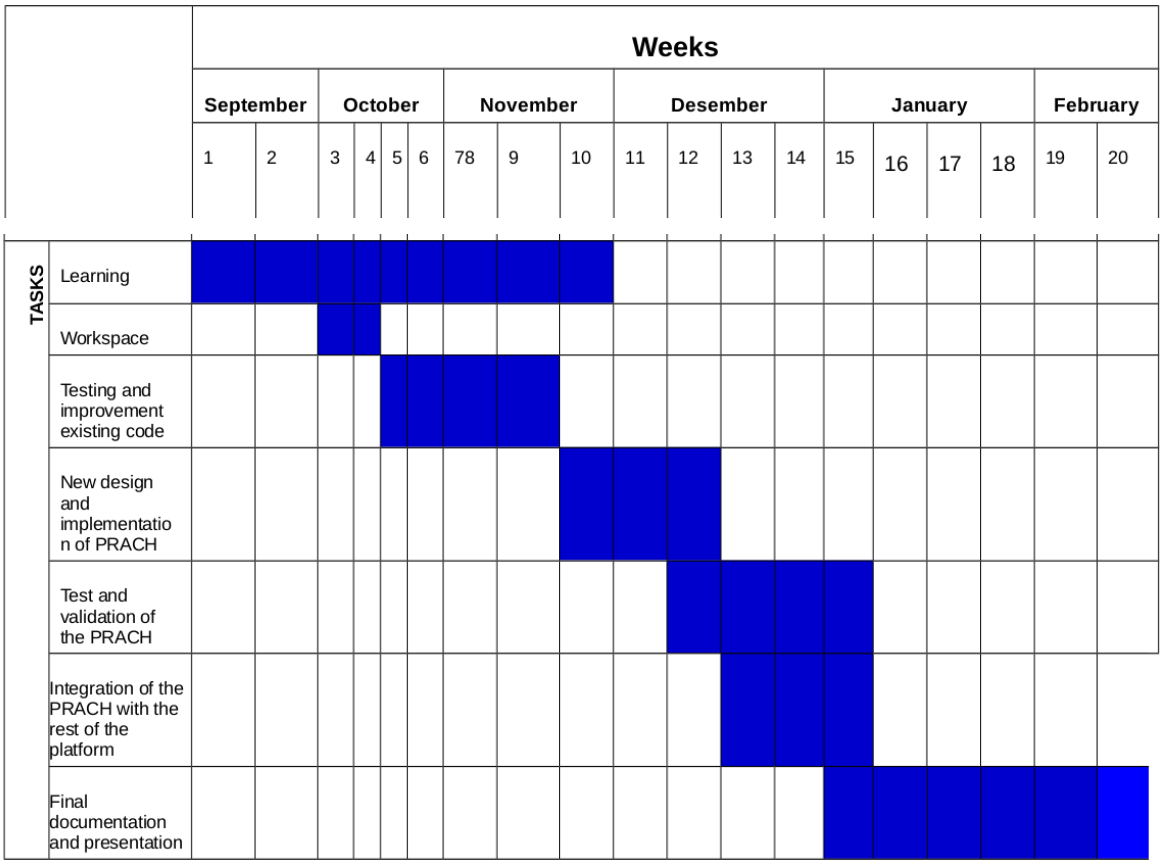
New design and implementation of PRACH Basing of on the limitation and problems detected during the testing task, design the new architecture of PRACH implementation and re-implement the algorithms.	Planned start date:02/11/2015 Planned end date: 02/12/2015	
	Start event: End event:	
Internal task T1: Re-implement the algorithms Internal task T2: Design of the of new PRACH software Internal task T3: Implementation	Deliverables:	Dates:

Project:PRACH implementation using SDR	WP ref: WP5	
Major constituent: Software	Sheet n of m	
PRACH Test and Validations Real test of PRACH and Validation of the final implementation	Planned start date: 03/12/2015 Planned end date: 15/12/2015	
	Start event: End event:	
Internal task T1: Test and Validation of PRACH	Deliverables:	Dates:

Project:PRACH implementation using SDR	WP ref: WP6	
Major constituent: Software	Sheet n of m	
Integration of PRACH with rest of the platform Integration the PRACH with other module of the base station.	Planned start date: 16/12/2015 Planned end date: 24/12/2015	
	Start event: End event:	
Internal task T1: integration of PRACH	Deliverables:	Dates:

Project:PRACH implementation using SDR	WP ref: WP7	
Major constituent: Software	Sheet n of m	
Final report Write the final report of the project and presentation.	Planned start date: 06/01/2016 Planned end date: 14/02/2015	
	Start event: End event:	
Internal task T1: Write the final report of the project Internal task T2: Write the presentation	Deliverables:	Dates:

Gantt Diagram



1.4. Deviations from the initial plan

Initially, the plan was to implement directly the RACH following the release 8 of LTE-UTMS. However, as the project went further, the existing implementation of the PRACH was observed to be useless since It was inadequately designed with a lot of bugs, complexity and resource management problems. So, it was decided that the PRACH should be implemented newly following the specifications and requirements above.

2. State of the art

In this chapter an overview of the concepts used in this project is done to enable a better comprehension. For that purpose, the RACH will initially be defined, its idea and functionality. Next, the PRACH, its structure, the mathematics behind it and its implementation will be given.

2.1. Random Access Channel

2.1.1. What is Random Access Channel

In LTE, the RACH is primarily used to achieve uplink synchronization between UE and eNodeB, and also for short message transmission. For time-synchronization, the RACH is normally used in the following scenarios [1].

- A UE in RRC_CONNECTED state, but not uplink-synchronized, needing to send new uplink data or control information (e.g an event-triggered measurement report).
- A UE in RRC_CONNECTED state, but no uplink-synchronized, needing to receive new downlink data, and therefore to transmit corresponding ACKnowledgement/Negative ACKnowledgement (ACK/NACK) in the uplink.
- A UE in RRC_CONNECTED state, handing over from its current serving cell to a target cell.
- For positioning purposes in RRC_CONNECTED state, when timing advance is needed for UE positioning.
- A transition from RRC_IDLE state to RRC_CONNECTED, for example for initial access or tracking area updates.
- Recovering from radio link failure.

2.1.2. Random Access Procedure

The LTE random access procedure comes in two forms, allowing access to be either contention-based (implying risk of collision) or contention-free.

A UE initiates a contention-based random access procedure in which a random access preamble signature is randomly chosen by the UE, enabling several Ues to simultaneously transmit the same signature, leading to a need for a subsequent contention resolution process. For the use-cases (from the scenarios where RACH occurred), which includes new downlink data and handover, the eNodeB has the option of preventing contention occurring by allocating a dedicated signature to a UE – a particularly important factor for the case of handover, which is time-critical.

A fixed number of 64 preambles signatures is available in each LTE cell, and the two type of PRACH procedures depend on a partitioning of these signatures between those for contention-based access and those reserved for allocation to specific UEs on a contention-free basis.

The two procedures are outlined in the following sections.

2.1.2.1. Contention-Based Random Access Procedure

The contention-based procedure consists of four-steps as shown in the Figure 2:

- Step1: Preamble transmission.
- Step 2: Random access response.
- Step 3: Layer 2/ Layer 3 (L2/L3) message.
- Step 4: Contention resolution message.

More details about the each step can be found in [1].

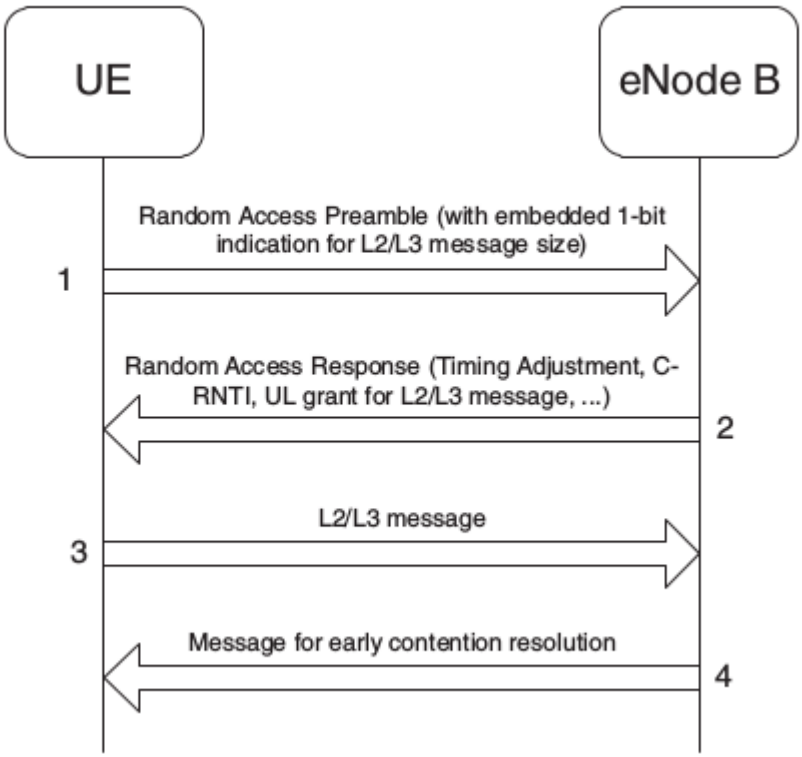


Figure 1. Contention-based Random Access Procedure

2.1.2.2. Contention-free Random Access Procedure

The slightly unpredictable latency of the random access procedure can be circumvented for some use-cases where low latency is required, such as handover and resumption of downlink traffic for a UE, by allocating signature to the UE on a per-need basis. In this case, the procedure is simplified to the following three-steps as we can see in Figure 2:

- Step 1: Preamble assignment.
- Step 2: Preamble transmission.
- Step 3: Random Access Response.

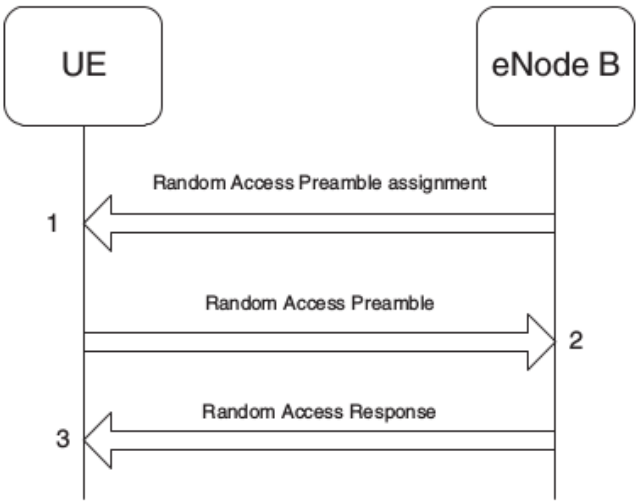


Figure 2. Contention-free Random Access Procedure

2.2. Physical Random Access Channel

The random access preamble part of the random access procedure is mapped at the physical layer onto the PRACH. A good understanding of the PRACH allows a better comprehension of the RACH.

2.2.1. Structure of Physical Random Access Channel

The LTE PRACH preamble consisted of a complex sequence which an OFDM symbol, built with a cyclic prefix (CP), thus allowing for an efficient frequency-domain receiver at the eNodeB. The preamble length is shorter than the PRACH slot in order to provide room for a guard time (GT) to absorb the propagation delay. See Figure 3.

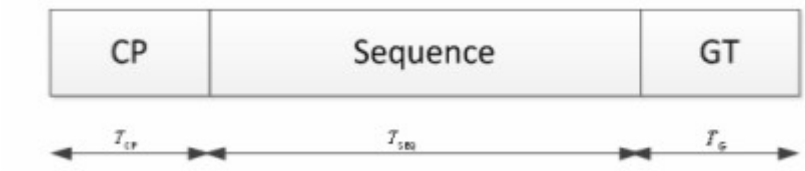


Figure 3. structure of Physical Random Access Channel preamble

2.2.1.1. Sequence Duration

The sequence duration T_{SEQ} is driven by the following factors:

- Trade-off between sequence length and overhead: a single sequence must be as as possible to maximize the number of orthogonal preambles, while still fitting a single subframe in order to keep the PRACH overhead small in most deployments.
- Compatibility with the maximum expected round-trip delay.

- Compatibility between PRACH and PUCH sub-carrier spacing.
- Coverage performance.

2.2.2. Physical Random Access Channel Formats

Four Random Access preamble formats are defined for FDD operation and one for TDD. Each format is defined by the duration of the sequence and its CP, as listed in figure 4 [2].

Preamble format	T_{CP}	T_{SEQ}
0	$3168 \cdot T_s$	$24576 \cdot T_s$
1	$21024 \cdot T_s$	$24576 \cdot T_s$
2	$6240 \cdot T_s$	$2 \cdot 24576 \cdot T_s$
3	$21024 \cdot T_s$	$2 \cdot 24576 \cdot T_s$
4*	$448 \cdot T_s$	$4096 \cdot T_s$

* Frame structure type 2 and special subframe configurations with UpPTS lengths $4384 \cdot T_s$ and $5120 \cdot T_s$ only.

Figure 4 Random access preamble formats

The preamble formats 0-3 correspond to frame structure type 1 which is applicable to FDD while preamble format 4 is for frame structure type 2, which is applicable to TDD. As you can see in the preview table, those parameters depend on the frame structure (Appendix 1) and the random access configuration which is controlled by higher layer.

As it can be seen in Figure 5 [1], there is multiple PRACH preamble formats for FDD due to wide range of the environment. However, this project focuses on TDD (format 4).

Preamble format	T_{CP} (μs)	T_{SEQ} (μs)	Typical usage
0	103.13	800	Normal 1 ms random access burst with 800 μs preamble sequence, for small to medium cells (up to ~14 km)
1	684.38	800	2 ms random access burst with 800 μs preamble sequence, for large cells (up to ~77 km) without a link budget problem
2	203.13	1600	2 ms random access burst with 1600 μs preamble sequence, for medium cells (up to ~29 km) supporting low data rates
3	684.38	1600	3 ms random access burst with 1600 μs preamble sequence, for very large cells (up to ~100 km)

Figure 5. Random Access preamble formats for FDD and it typical usage

2.2.3. Preamble Sequence Theory and Design

In LTE prime-length Zadoff-Chu (ZC) sequences have been chosen to generate the preamble sequence. These sequence enable improved PRACH preamble detection performance. In particular:

- The power delay profile is build from periodic instead of aperiodic correlation.
- The intra-cell interference between different preamble received in the same PRACH ressource is reduced.
- Intra-cell interference is optimized with resprect to cell size: the smaller the cell size, the larger the number of orthogonal signatures and the better the detection performance.
- The eNodeB complexity is reduced.
- The support for high-speed UE is improved.

2.2.3.1. Zadoff-Chu Sequences

ZC sequences are non binary unit-amplitude sequence, which satisfy a Constant Amplitude Zero Autocorrelation (CAZAC) property. CAZAC sequences are complex signals of form $e^{j\alpha_k}$. The ZC sequence of odd-length N_{ZC} is given by

$$\alpha_q(n)=\exp\left(-j2\pi\frac{\frac{n(n+1)}{2}+nl}{N_{Zc}}\right),$$

$$where: q=0,1,\ldots\ldots,N_{ZC}\ is\ ZC\ sequence\ root\ index,$$

$$n=0,1,\ldots\ldots N_{ZC},$$

$$l=any\ integer.\ But\ LTE\ we\ take\ l=0\ for\ simplicity.$$

ZC sequences have the following three important properties:

Property 1: A ZC sequence has constant amplitude, and its l point DFT also has constant amplitude. The constant amplitude property limits the Peak-to-Average Power Ration (PAPR) and generates bounded and time-flat interference to other users. It also simplifies the implementation as only phases need to be computed and stored, not amplitudes

Property 2: ZC sequences of any length have an 'ideal' cyclic autocorrelation (i.e the correlation with its circularly shifted version is delta function). The zero autocorrelation property may be formulated as:

$$r_{kk}=\sum_{n=0}^{N_{zc}-1}\alpha_q(n)\alpha_q[(n+\sigma)]=\delta(\sigma),$$

where $r_{kk}(.)$ is the discrete periodic autocorrelation function of α_q at lag σ. This property is of major interest when the received signal is correlated with a reference sequence and the received reference sequences are misaligned.

Property 3: The absolute value of the cyclic cross-correlation function between any two ZC sequences is constant and equal to $1/\sqrt{N_{ZC}}$ if |q1 – q2| (where q1 and q2 are the

sequence indices) is relatively prime with respect to (a condition that can be easily guaranteed if N_{ZC} is a prime number) the cross-correlation of sqrt(N_{ZC}) at lags achieves the theorical minimum cross-correlation value for any two sequences that have ideal autocorrelation. Selecting N_{ZC} as prime number results in (N_{ZC}) ZC sequence that have the optimal cyclic cross-correlation between any pair. However, it is not always convenient to use sequences of prime length. In general, a sequence of non-prime length may be generated by either cyclic extension or truncation of a prime-length ZC sequence.

A further useful property of ZC sequences is that DFT of a ZC sequence $\alpha_q(n)$ is a weighted cyclically shifted ZC sequences $X_x(k)$ such that $w = -1/q \bmod N_{ZC}$. This means that ZC sequence can be generated directly in the frequency domain without the need for a DFT operation.

2.2.3.2. Preamble sequence duration

The sequence length design should address the following requirements:

- Maximize the number of ZC sequences with optimal cross-correlation properties.
- Minimize the interference to/from the surrounding scheduled data on the PUSHC.

The former requirement is guaranteed by choosing a prime length sequence. For the latter, since data preamble OFDM symbols are neither aligned nor have the same durations, strict orthogonality cannot be achieved. At least, fixing the preamble duration to an integer multiple of PUSHC symbol provides some compatibility between preamble and PUSCH subcarriers. However, with $800\,\mu s$ duration, the corresponding sequence length would be 864, which does not meet the prime number requirement. Therefore, shortening the preamble to a prime length slightly increases the interference between PUSCH and PRACH by slightly decreasing the preamble sampling rate.

The PRACH uses guard bands to avoid the data interference preamble edges. A cautious design of preamble sequence length not only retains a high inherent processing gain, but also allows avoidance of strong data interference. In the absence of interference, it can be seen that reducing the sequence length below 839 gives no further improvement in detection rate. Therefore the sequence length 839 is selected for LTE PRACH (FDD case).

For TDD case, applying the approach explained for FDD, the length sequence of 139 is selected.

2.2.3.3. Preamble sequence generation

As said before and illustrated in [2], there are 64 preambles available in each cell. The set of 64 preamble sequences in a cell is found by including first, in the order of increasing cyclic shift, all the available cyclic shifts of root ZC sequence with logical index parameter (Appendix 3) provided by higher layer. Additional preamble sequences, in case 64 preambles cannot be generated from a single root ZC sequence, are obtained from the root ZC sequences with consecutive logical index (from 0 to 837) until de 64 preamble are found. Here it should noted once more that all the 64 preamble sequences are generate in frequency domain (due to ZC properties).

The u^{th} root ZC sequence is defined by

$$x_u(n) = e^{j\frac{\pi u n(n+1)}{N_{ZC}}}, 0 \le n \le N_{ZC} - 1,$$

where the length N_{ZC} of the ZC sequence is 839 for FDD and 139 for TDD.

From the u[th] root ZC sequence , RA preambles with zero correlation zone of length Ncs – 1 are defined by cyclic shifts according to

$$x_{u,v}(n) = X_u\left((n + C_v) \bmod N_{ZC}\right),$$

where the cyclic shift is given by

$$C_v = \begin{cases} vN_{cs} & v = 0,1\ldots,\lfloor N_{ZC}/N_{CS}\rfloor - 1, N_{cs} \neq 0 \quad \textit{for unrestricted sets.} \\ 0 & N_{cs} = 0 \quad \textit{for unrestricted sets.} \\ d_{start}\lfloor v/n_{shift}^{RA}\rfloor + \left(v \bmod n_{shift}^{RA}\right)N_{cs} & v = 0,1\ldots,n_{shift}^{RA}n_{group}^{RA} + \overline{n_{shift}}^{RA} - 1 \quad \textit{for restricted sets.} \end{cases}$$

and N_{CS} values can be found in Appendix 4. The parameter high-speed flag given by the higher layer determines if unrestricted set or restricted set shall be used.

As noticeable in the previous formula, if the preamble is using unrestricted set, it is pretty simple to find C_v, we only need to know N_{CS} and N_{ZC}. The problem is when the preamble is using the restricted set, where we first need to know first to find d_u which is the cyclic shift corresponding to a Doppler shift of magnitude $1/T_{SEQ}$, which is given by

$$d_u = \begin{cases} p & 0 \le p < N_{zc}/2 \\ N_{ZC} - p & otherwise \end{cases},$$

where p is the smallest non-negative integer that fulfils $(pu) mod N_{Zc} = 1.$

Once we have d_u , we can them compute the rest of the parameter necessary for computing C_v for restricted set.

So for $N_{cs} \le d_u < N_{ZC}3$, we have:

$$n_{shift}^{RA} = \lfloor d_u/N_{cs}\rfloor,$$
$$d_{start} = 2d_u + n_{shift}^{RA}N_{cs},$$
$$n_{group}^{RA} = \lfloor d_u/d_{start}\rfloor,$$
$$\overline{n_{shift}}^{RA} = max\left(\lfloor\left(N_{ZC} - 2d_u - n_{shift}{}^{RA}d_{start}\right)/N_{cs}\rfloor, 0\right),$$

and for $N_{cs}/3 \le d_u \le \left(N_{zc} - Ncs\right)/2$, they are:

$$n_{shift}^{RA} = \lfloor\left(N_{ZC} - 2d_u\right)/N_{CS}\rfloor,$$
$$d_{start} = N_{ZC} - 2d_u - n_{shift}^{RA}N_{cs},$$
$$n_{group}^{RA} = \lfloor d_u/d_{start}\rfloor,$$
$$\overline{n_{shift}}^{RA} = max\left(\lfloor\left(N_{ZC} - 2d_u - n_{shift}{}^{RA}d_{start}\right)/N_{cs}\rfloor, n_{shift}^{RA}\right),$$

2.2.3.4. Baseband signal generation

In order to transmit the preamble sequence generated in the frequency domain, we have to convert this data into a time domain sequence. For that, [2] give the following expression:

$$S(t)=\beta_{PRACH}\sum_{k=0}^{N_{ZC}-1}\sum_{n=0}^{N_{ZC}-1}x_{u,v}(n)\cdot e^{-j\frac{2\pi nk}{N_{ZC}}}\cdot e^{j2\pi(k+\phi+K(k_0+12))\Delta f_{RA}(t-T_{CP})},$$

where:

$0\le t<T_{SEQ}+T_{CP}$,

β_{PRACH} is the amplitude scaling factor,

$k_0=n_{PRB}^{RA}N_{sc}^{RB}-N_{RB}^{UL}N_{sc}^{RB}/2$,

where n_{PRB}^{RA} is the location in frequency domain and it is can be expressed as:

$$0\le n_{PRB}^{RA}<N_{PRB}^{UL}-6,$$

N_{PRB}^{UL} is the uplink system bandwidth (in Resource Block (RB)).

N_{SC}^{RB} Is the number of sub-carriers per RB and is 12.

$k=\Delta F/\Delta f_{RA}$ is the factor that accounts for ratio of sub-carrier spacing between PUSCH and PRACH.

ϕ is a fixed offset.

Both k and ϕ determine the frequency-domain location of the random access preamble within the physical resource blocks and their values can be found in Appendix 5.

2.3. Physical Random Access Channel Implementation

2.3.1. UE Transmitter

For the generation and transmission of the PRACH preamble, the process shown in Figure 6 was followed. This process represents the baseband signal discussed in chapter 2.2.3.4.

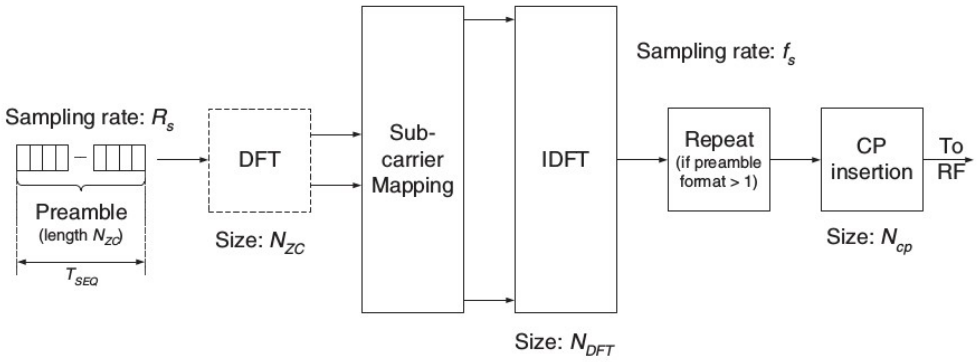


Figure 6. Functional structure of PRACH preamble transmitter

The DFT operation after the preamble sequence has been generated to reduce the PAPR and the IDFT operation is to do SC-FDMA modulation [5]. The cyclic shift can be implemented either in the time domain after the IDFT, or in the frequency domain before the IDFT through a phase shift.

2.3.2. enodeB PRACH receiver

The whole procedure implemented as PRACH receiver in the eNodeB, can divided in three steps [1] shown in the Figure 7:

- CP removal.
- Computation of PDP.
- Signature detection.

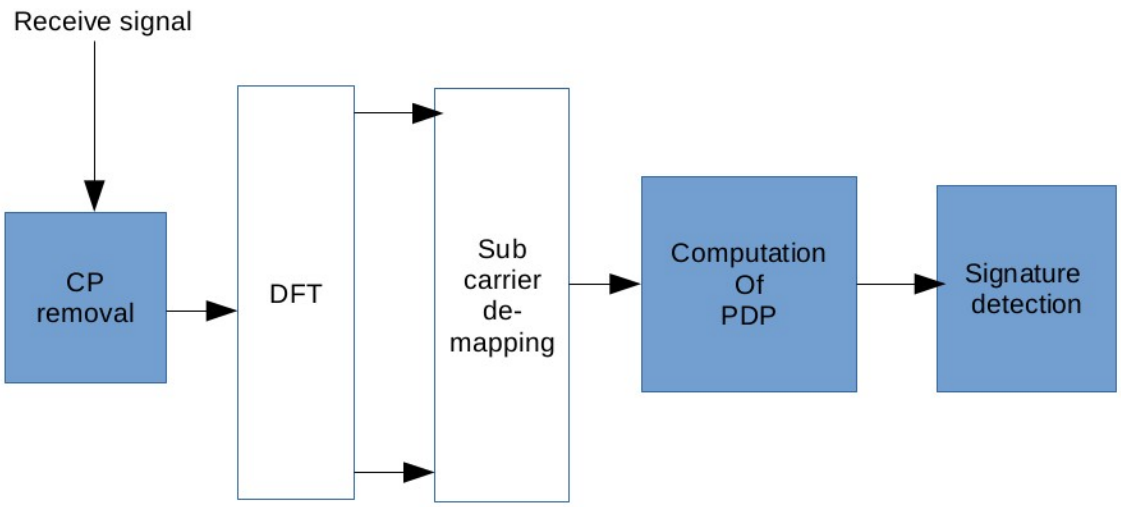


Figure 7: PRACH receiver steps

The CP removal block consists on removing the CP add by the UE to protect the PRACH sequence against inter-symbol interference (ISI) during the transmission.

2.3.2.1. Computation of Power Delay profile

This step is a significant part of the receiver before the preamble detection is done. It based on the correlation algorithm between the received sequence and local ZC root sequences. Since the correlation can be computed by the convolution operation in time domain or the multiplication in frequency domain, we implemented this step using the frequency domain correlation calculation following the process illustrated in Figure 8:

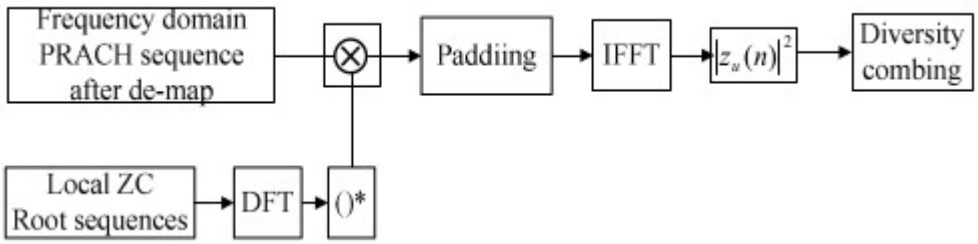


Figure 8: Computation of Power Delay Profile process

After cyclic prefix removal and FFT, the frequency domain PRACH sequences are de-mapped from the corresponding time frequency resource blocks. Next, the frequency domain correlation with the local ZC root sequences. For that local ZC root sequences are transformed to the frequency domain by the DFT module and the conjugate operation is performed on the corresponding frequency sequence. Then, the frequency correlation is calculated as follows:

$$Z_u(k) = Y(k)\, X_u^*(k),$$

where the $X_u(k)$ is the DFT of a local ZC root sequence, $Y(k)$ is the DFT of the received preamble sequence and $Z(k)$ is the frequency correlation result. Since we have seen that there are commonly 64 preamble sequences available in each LTE cell, and that we need more than one ZC root sequences to produce all the preamble sequence, the local ZC root sequence used for the frequency correlation are the one used to generate the all 64 preamble sequences.

The padding module aims at providing the desired oversampling factor and or adjusting the resulting number of samples to a convenient IFFT size.

Finally, the PDP of the correlation defined in [6] is computed as:

$$z_p(l) = |z_l|^2.$$

We first need to find time domain correlation by using of the IFFT module.

In the last module, we combine all the results from different antennas to take advantage of the diversity gain.

2.3.2.2. Signature Detection

As explain in [6], the detection of the preamble required the computation of the PDP. For the detection we used two thresholds (Figure 9), the first one (thresholdA) for the detection of the arrival signal which determines whether the search window need to proceed the peak detection, and the second one (thresholdB) for the peak detection which determines the existence of an access. The process is implemented as follows:

- A ZC root sequence can generate several preamble sequences by cyclic shifts (discussed in chapter 2.2.3.3), that means that the frequency-domain computation of PDP of a root sequence provides in one shot the concatenated PDPs of all signatures derived from the same root sequence. So, by separating the length of the PDP to several search windows, the search process and the detection can be performed in every window. The length of the search windows is:

$$W = N_{CS} \cdot N_{IFFT} N_{ZC}.$$

- Estimation of the noise level, which can be computed according to PDP.
- Set up the threshold of arrival signal (thresholdA). If in search window, the power level is below than thresholdA, the receiver is aware of no preamble's arrival in this window.
- Set up the for detection of existence of an access (thresholdB). If the peak power of the window is above thresholdB, then the receiver make a decision of a preamble's access in the current window.

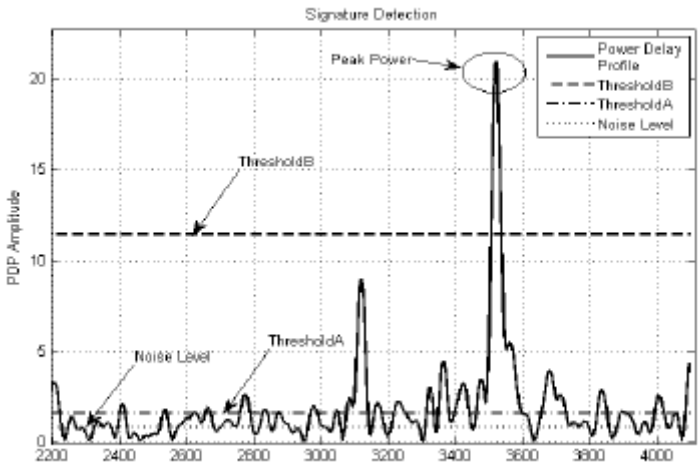


Figure 9. Detection Threshold A and B

The explications about how to estimate the noise level and set up the thresholds A and B can be found in the Appendix 6.

3. Project development:

This chapter summarizes the practical phase of the project. First, the code of the PRACH module implementation is discussed. Next, the code for the integration of the PRACH module incorporated in the rest of the system will be discussed. All the system is built using C++ programming language.

3.1. PRACH Module

The previous implementation of the PRACH was not very practical, it presented many problems of software development (Bad design, no object oriented, bad presentation etc...) which makes it not suitable for the integration due to algorithm inefficiency (e.g. memory leak).

With the goal to deal with the problem of design following the object oriented paradigm, the architecture of the new PRACH module is shown in the UML represented below:

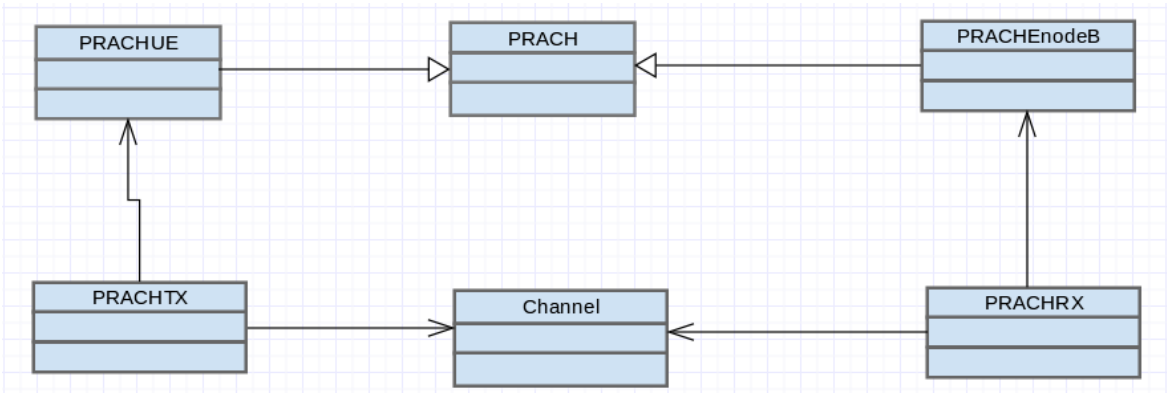


Figure 10. UML of PRACH

This UML shows that the implementation can be divided in three parts:

- The implementation of the PRACH sequence.
- Implementation of PRACH baseband.
- Implementation of PRACH baseband receiver.

3.1.1. PRACH Sequence

The implementation of the PRACH sequence generator includes three classes: PRACH, PRACHENODEB, PRACHUE.

PRACH

This is an abstract class (interface in OOP), which means that it cannot be instantiated directly. It defines the core functions necessaries for generation of a PRACH sequence such as the generation of the ZC root sequence and the generation of cyclic shift. It also manages the FFT and IFFT of the PRACH sequence generated.

PRACHEnodeB

This class represents a particular implementation of PRACH sequence generator in the eNodeB side. It ensures a proper set-up of the followings parameters necessaries to generate the PRACH sequence:

- prachConfigIndex: determines what type of preamble should be used and at which system frame and suframe UE can transmit PRACH preamble. It values can be found in Appendix 6.
- zeroCorrelationZoneConfig and Highspeedflag: determine the cyclic shift intervals (N_{CS}) to generate the 64 PRACH sequences (Appendix 4).
- prachFreqOffset.
- rootSequenceIndex: determines the first root sequence number used to generate ZC sequence (Appendix 3).

It generates all the 64 PRACH sequences available for a cell and their respective FFT to speed up the the baseband processing.

PRACHUE

This class is quite similar to PRACHENodeB class due to the fact that its major purpose is to manage the generation of the PRACH sequence in the UE. Unlike the PRACHENodeB, it can only generates one PRACH sequence at a time only for the index that the UE has selected. Moreover, it manages the configuration parameter of the PRACH differently.

3.1.2. PRACH sequence Transmitter

The PRACH sequence transmitter in UE is represented by PRACHTX class. This class takes care of the UE parameters set-up broadcasted by eNodeB (PRACH configuration parameters, cell identification etc...), manages the generation of the baseband sequence and passes it to the next module for its transmission.

The implementation of the baseband generation code is as illustrated in Figure 6. First the PRACHUE class to generate the PRACH sequence, next the FFT and IFFT are performed to generate the baseband and finally the CP is added to obtain the symbol sequence transmitted

3.1.3. PRACH sequence Receiver

PRACHRX class represents the PRACH sequence receiver in LTE base station. Using the PRACHEnodeB class, it sets-up the PRACH parameters, generates the 64 PRACH sequence and is responsible to receive the baseband sequence, processes it in order to detect the PRACH sequence transmitted.

The implementation follows what has been explained in chapter 2.3.2 except the signature detection, where the threshold (it detects when there is preamble transmission) is set-up empirically.

3.1.4. Code Optimization

To improve the performance of PRACH module, an emphasis is made on the optimization of the memory management and execution time in the baseband processing.

The execution time is very crucial in the baseband processing since the eNodeB receiver only has one subframe (1ms) to detect the PRACH sequence transmitted. Moreover, it also interesting that the eNodoB could answer in a constant time.

For memory management, it is important to minimize the memory used in the PRACH module due the limitations of the device where the final implementation will be running. The main problem here is memory leak cause by non-free dynamic memory (memory allocated outside the stack). To solve those problem, one crucial point in the implementation is the way the FFTW library is used. The FFTW library is used for optimal computation of FFT and IFFT in C/C++ programming . To understand this approach, it is important to understand how the library works.

As explained in [7], computing the FFT/IFFT using FFTW library, the process works as follows:

- Create a plan (using the input data size and the size of the desired output data) which is special space where the computation of FFT is optimized. This process dynamically allocated memory.
- Compute the FFT/IFFT by executing the plan created.
- Destroy the plan.

The fact of creating and destroying the plan for each FFT or IFFT needed decreases the performance of the PRACH implementation. To improve this, the approach adopted can be summarized as follows:

- Look for a plan which is prepared for the FFT/IFFT to computed or create a new one if no plan has been found.
- Copy the PRACH sequence whose FFT/IFFT is going to be computed in the plan
- Execute the plan to compute the FFT/IFFT.
- Copy the result out the plan.

As it can be observed, The plan can be destroyed only when the PRACH module is ready. That way, this will only create a new plan FFT/IFFT calculation when there is no existing plan which is prepared for the current input data and desired output. This enables the reduction of the cost for generating FFT/IFFT because most of the time, Only copy operations are made.

Nevertheless, this approach increases the memory-leak problems, but this can be solved easily by making sure that all the dynamic memories allocated are freed at the end of PRACH module.

To speed up the baseband processing and achieve constant time response, the following things are done in the:

- For all the cases known, all the dynamic memory are preallocated to avoid doing during the baseband processing.

- Pre-calculation of all the PRACH FFT.

3.2. Integration of PRACH module in the system

The system is implemented so that for uplink transmission, the UE generates all the information that needs to transmit in the resource manager module and passes it to pipe module for its transmission. In the other hand, the eNodeB receives the information transmitted using the resource manager and passes it to one of the prepared pipe for further processing.

As shown in the figure 10.1, the integration of PRACH module is done such that in the UE side, PRACH and baseband sequences are handled in the resource manager, next the baseband sequence is passed to pipe module for its transmission. In the case of the eNodeB, the resource manager takes care of the baseband reception and the prepared pipe of the baseband processing and PRACH sequence detection.

The Figures 10.2 and 10.3 illustrated the UML of how the PRACH code was integrated in the UE side as well as in the eNodeB side. In the UE, the resource manager is represented by the UEController class which uses the PRACHTX class to generate the PRACH and baseband sequence. The pipe in the other hand, is represented by LTETXRelease class and the resource manager can communicate with it using LTETXInfo class.

In the case of eNodeB, the pipe is represented by the LTERXRelease class which uses the PRACHRX class to manage the baseband processing and the PRACH sequence detection. The resource manager is represented by the MACScheduler class. The communication between resource manager and pipe is achieved using LTERXInfo.

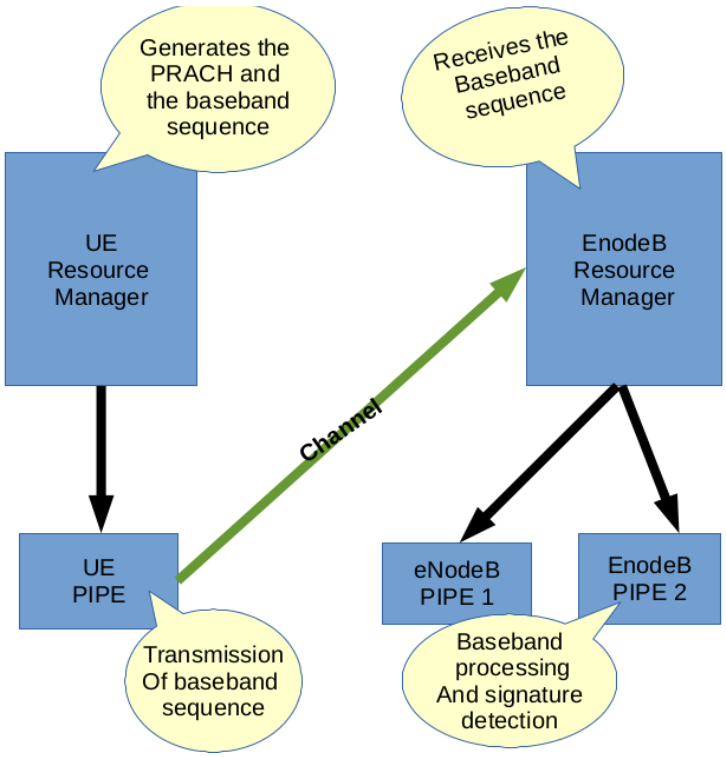


Figure 10.1. Integration of PRACH module

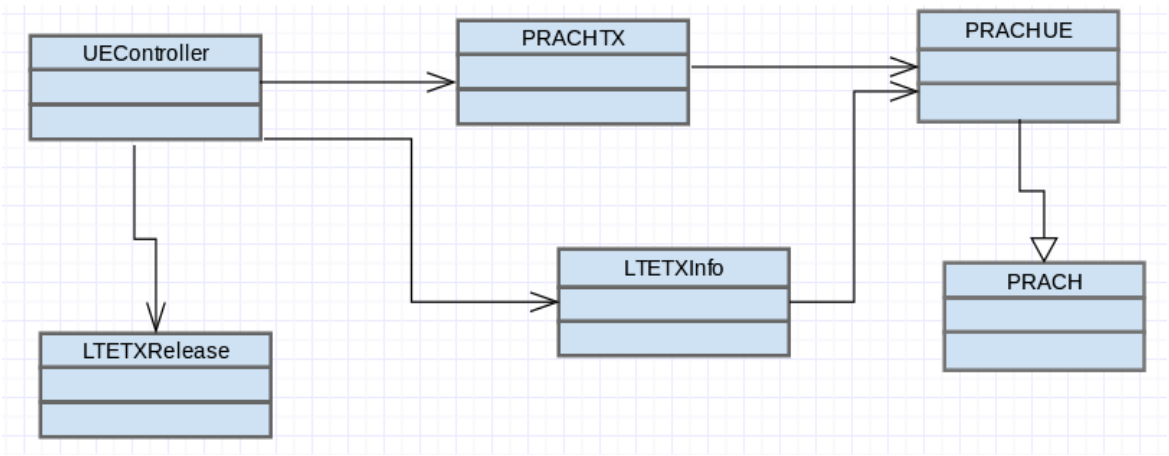


Figure 10.2. UML for integration of PRACH in UE side

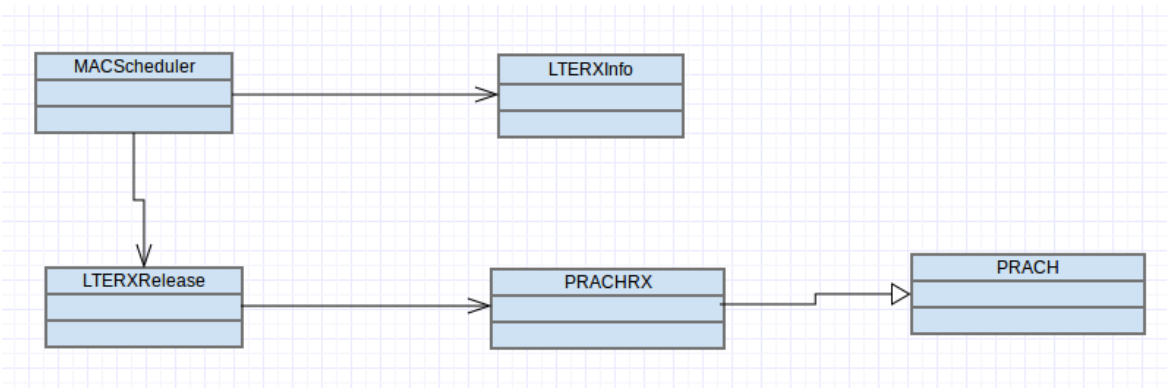


Figure 10.3 UML of PRACH in eNodeB side

4. Simulation and Results

This chapter talks about the tests done to verify the implementations. For all the tests written, scenarios where with only one UE and one eNodeB were considered.

4.1. Simulation environment

To run the tests, the working environment is set up as follows:

- A Computer with GNU/Linux installed. GNU/Linux is a free and open-source operative system developed by Linus Torvalds and Free Software Foundation.
- Installation of GNU Radio: a free and open-source software development toolkit that provides signal processing blocks to implement software radio.
- Installation of G++ compiler: a free and open-source compiler for c++ programming language.
- Installation of Google Test (gtest): a unit testing library for the C++ programming language.

4.2. PRACH module

To verify the PRACH implementation, a scenario with a TDD-LTE eNodeB, and UE with low speed were considered. Three tests were prepared.

Generation of PRACH sequences

The purpose of this test is to ensure that the eNodeB can generate correctly 64 PRACH sequences available. For that, the followings things are checked:

- eNodeB is able to generate exactly 64 PACH sequences.
- PRACH sequences from the same group (those generated using same ZC sequence and different cyclic shifts) are orthogonal. This can be done showing the zero autocorrelation property which means that the result of correlation between PRACH sequences of the group is 0. Figure 11 shows the result of the calculation of the correlation between the first PRACH sequence of a group with the rest of the group sequences. The zero autocorrelation property can clearly be seen.

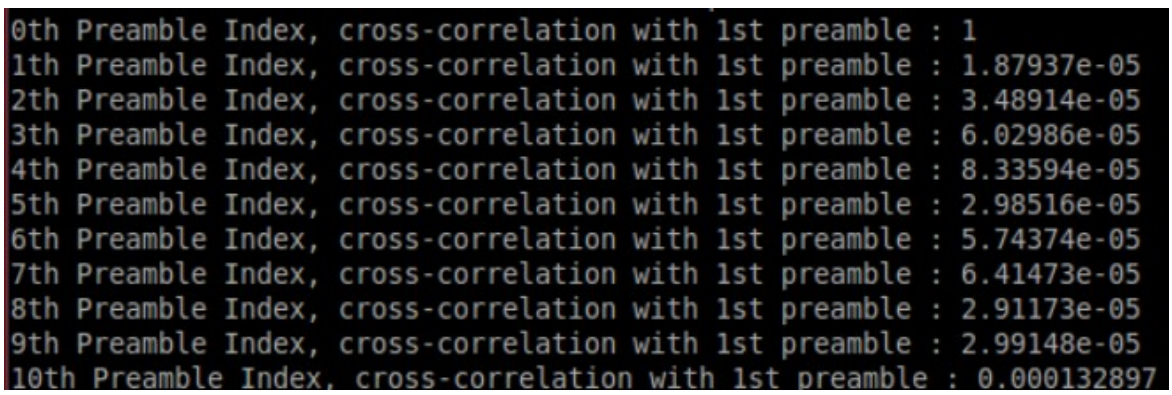


Figure 11. PRACH sequence Zero Autocorrelation property

- Multiple ZC root sequences are used to generate the 64. The Figure 12 shows the result of correlation between the first PRACH sequence and the rest. As we see, from the 2nd to 10th preamble index, the result is closed to 0 which means that the belongs to same the group than the 1st (zero autocorrelation property). But from 11th to the end, there is no zero autocorrelation property, this shows that those PRACH sequences belong to a different group.

```
0th Preamble Index, cross-correlation with 1st preamble : 1
1th Preamble Index, cross-correlation with 1st preamble : 1.87937e-05
2th Preamble Index, cross-correlation with 1st preamble : 3.48914e-05
3th Preamble Index, cross-correlation with 1st preamble : 6.02986e-05
4th Preamble Index, cross-correlation with 1st preamble : 8.33594e-05
5th Preamble Index, cross-correlation with 1st preamble : 2.98516e-05
6th Preamble Index, cross-correlation with 1st preamble : 5.74374e-05
7th Preamble Index, cross-correlation with 1st preamble : 6.41473e-05
8th Preamble Index, cross-correlation with 1st preamble : 2.91173e-05
9th Preamble Index, cross-correlation with 1st preamble : 2.99148e-05
10th Preamble Index, cross-correlation with 1st preamble : 0.00013289
11th Preamble Index, cross-correlation with 1st preamble : 0.0847894
12th Preamble Index, cross-correlation with 1st preamble : 0.0848002
13th Preamble Index, cross-correlation with 1st preamble : 0.0848547
14th Preamble Index, cross-correlation with 1st preamble : 0.0848243
15th Preamble Index, cross-correlation with 1st preamble : 0.0848856
16th Preamble Index, cross-correlation with 1st preamble : 0.0848394
17th Preamble Index, cross-correlation with 1st preamble : 0.0848488
18th Preamble Index, cross-correlation with 1st preamble : 0.0848277
19th Preamble Index, cross-correlation with 1st preamble : 0.0848445
20th Preamble Index, cross-correlation with 1st preamble : 0.084819
21th Preamble Index, cross-correlation with 1st preamble : 0.084822
22th Preamble Index, cross-correlation with 1st preamble : 0.0848686
23th Preamble Index, cross-correlation with 1st preamble : 0.0849427
24th Preamble Index, cross-correlation with 1st preamble : 0.0847107
25th Preamble Index, cross-correlation with 1st preamble : 0.0847539
26th Preamble Index, cross-correlation with 1st preamble : 0.0847379
27th Preamble Index, cross-correlation with 1st preamble : 0.0848472
28th Preamble Index, cross-correlation with 1st preamble : 0.0848135
29th Preamble Index, cross-correlation with 1st preamble : 0.0847776
30th Preamble Index, cross-correlation with 1st preamble : 0.0848859
31th Preamble Index, cross-correlation with 1st preamble : 0.0849031
32th Preamble Index, cross-correlation with 1st preamble : 0.0848328
33th Preamble Index, cross-correlation with 1st preamble : 0.0848682
34th Preamble Index, cross-correlation with 1st preamble : 0.084781
35th Preamble Index, cross-correlation with 1st preamble : 0.0847956
36th Preamble Index, cross-correlation with 1st preamble : 0.0847842
37th Preamble Index, cross-correlation with 1st preamble : 0.0848337
38th Preamble Index, cross-correlation with 1st preamble : 0.0848003
```

Figure 12. Different groups demonstration

- The PRACH sequence generated are the expected one. For that, firstly using the same PRACH configuration parameters, 64 PRACH sequence were generated with both Matlab simulator and the COMNET PRACH module. Next, the correlation was computed for comparison purpose. As shown in Figure 13, the PRACH sequence generated with COMNET PRACH module are identical to those generated with Matlab code given that the result is always one for all the cases.

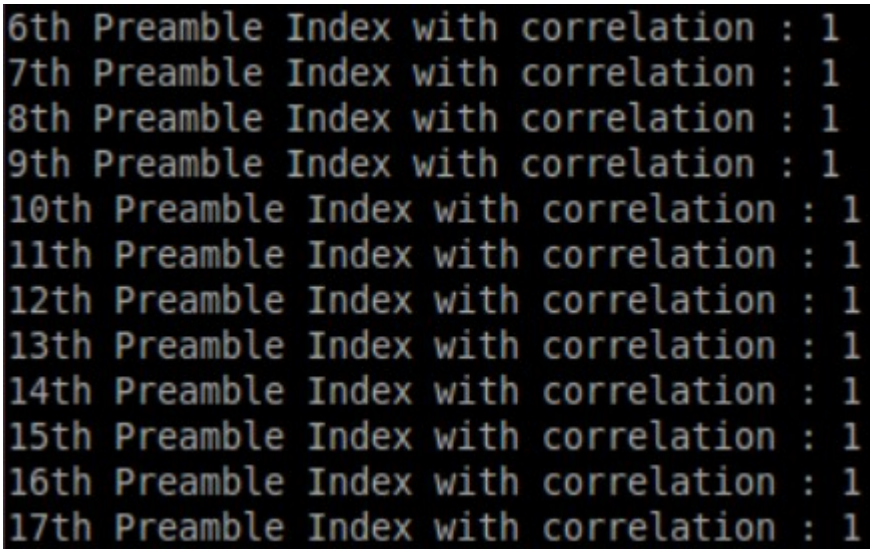


Figure 13. PRACH sequence comparison

Generation of baseband sequence

In this test, it was ensured that the UE is able generate the PRACH sequence and the baseband sequence. To do that, firstly, a baseband signal for a specific PRACH sequence with Matlab code and the COMNET PRACH module, next the cumulative error was computed in order to see the difference. In the Figure 14, it can clearly be seen that the result of the test is almost zero.

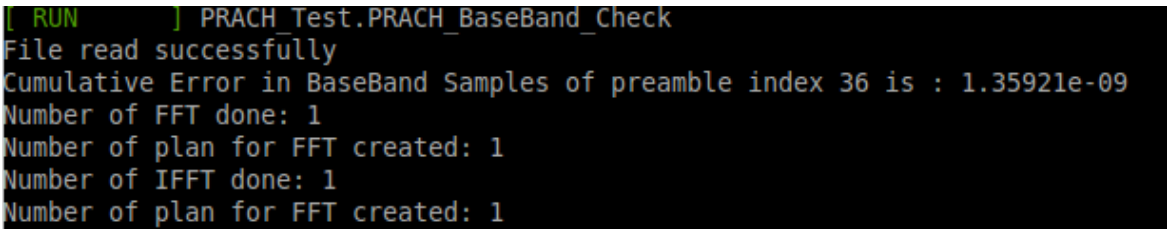


Figure 14. Baseband test result

Baseband processing

Assuming that there is no noise in the channel and the eNodeB receives exactly what the UE transmitted, this test checks a full functionality of PRACH module, for that the following points are checked:

- UE and eNodeB can be set up correctly.
- eNodeB generates 64 PRACH sequences.
- UE can generate a baseband sequence for a PRACH index assigned and transmit it.
- The eNodeB can process the baseband signal received and detect the PRACH sequence transmitted.
- The baseband sequence processing is done in approximately 1ms.

Figure 15 and 16 show the result of the test of the baseband processing. In this case it can be observed that the test lasts 2 ms. This is due to the fact that the test includes UE and eNodeB set up; generation of the 64 PRACH sequences for eNodeB; baseband generation and transmission by the UE; baseband reception and processing in the eNodeB. This means that if all those additional operations are removed from the test, the the baseband processing can be done in less than on 1ms. It should also be noted that those 2 ms is the best result achieved. The rest of the cases were always 3 ms.

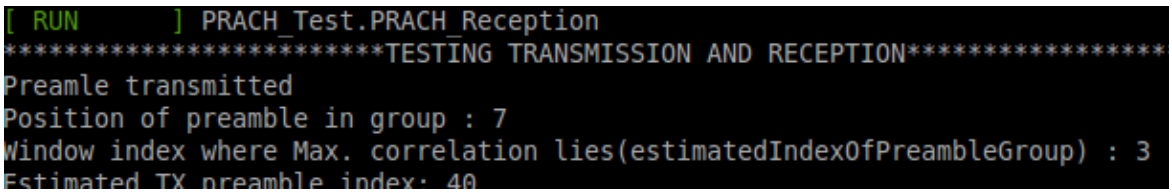


Figure 15. PRACH reception test result

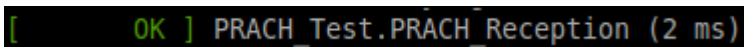


Figure 16. PRACH reception time duration

4.3. PRACH module Integration

For the PRACH module integration incorporated in the system, the following were tested:

- UE resource manager is able to generate the baseband signal, add to the work item and pass it to pipe for transmission in the subframe assigned by the eNodeB.
- The eNodeB resource manager is able to receive the work item with baseband signal, passes it to the prepared pipe for baseband processing and PRACH sequence detection.

Figure 17 shows the simulation environment for PRACH Integration with one eNodeB and UE set up. Figure 18 shows the generation of the PRACH sequence and its transmission in the sub-frame 2 while in Figure 19, the reception and the detection of PRACH sequence are shown. This is another proof that PRACH sequence receiver is able to retrieve the PRACH sequence transmitted in approximately 1 ms.

Figure 17. PRACH integration test set up on simulator

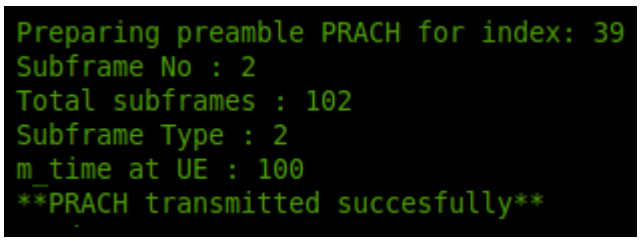


Figure 18. PRACH sequence transmission in PRACH integration test

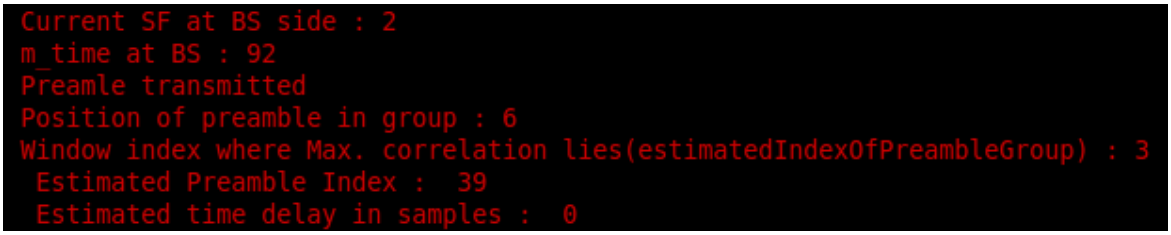


Figure 19. PRACH sequence reception in PRACH integration test

5. Budget

Given that neither physical component nor prototype was designed for this project, this budget was estimated taken into account the number of hours dedicated to the research developing the software and the planning simulation case.

Furthermore, the only software tool license considered is MATLAB license (since the rest (GNU Linux, GNURadio etc...) are released with free and open-source license), the computer used for the development of the project. Therefore, we have:

- Total hours estimated: 800h. Considering salary a junior software developer engineer (8€/h), this is 6400€.
- Matlab license: 1000€.
- Eight core desktop computer AMD FX 8350: 900€.

The final budget dedicated to the project is 8300€.

6. Conclusions and future development

PRACH module is critical to deal with time synchronization in uplink transmission between the UE and eNodeB for TD-LTE platform COMNET is implementing. In this project, the design and implementation of the PRACH module have been achieved. Some of the main tasks done are:

- Implementation of the PRACH sequence generator.
- Implementation of the PRACH manager for the UE (generation of PRACH sequence baseband signal).
- Implementation of PRACH manager for the eNodeB (generation of all 64 PRACH sequence available for a cell, baseband processing and PRACH sequence detection).
- Correct incorporation of the PRACH module in the rest of the system.

Even though the PRACH module implemented satisfies the requirement set up and works fine in simulation test, for the real case testing, it still important to implement the threshold used in the baseband processing for signature detection correctly. Right now the threshold used is prepared for simulation test, but it cannot be guaranteed that it will work correctly in a real test since we need to consider signal to noise ration (SNR).

Furthermore, in the PRACH integration, we need to implement an algorithm to determine correctly when to transmit the PRACH sequence. In my current implementation, we have fixed it to a specific subframe (e.g subframe 2).

Bibliography:

[1] Stefani Sesia, Issam Toufik, Matthew Baker. Faludi. *LTE: The UMTS Long Term Evolution: From Theory to Practice, 2nd ed. John Wiley & Sons Ltd, 2011.7*

[2] Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulations. 3GPP TS 36.211, version 8.7.0, Release 8.

[3] RACH, available: http://www.sharetechnote.com/html/RACH_LTE.html

[4] Lee, Joohyun, “Implementation of low latency Random Access Detector for 4G Cellular System” Ph.D. dissertation, Department of Information and Communications Engineering, KAIST, Korea, 2013.

[5] PENG Feilong, Wang Weidong "A high estimated accuracy Random Access Preamble Detection Algorithm for FDD-LTE system".

[6] Yanchao Hu, Juan Huan, Huajie Gao, Yongtao Su, Jinglin Shi. "A Method of PRACH detection Threshold Setting in LTE TDD Femtocell System". *In Proceedings of the seventh International ICST conference on Communication and Networking, 2012*, China. pp. 409-411.

[7] FFTW, available: http://www.fftw.org/fftw3_doc/Complex-One_002dDimensional-DFTs.html#Complex-One_002dDimensional-DFTs.

Appendix 1: Physical Random Access Channel Frame Structure

As specified in [2], The transmission of a random access preamble is organized into radio frame with $T_f = 307200 \times T_s = 10ms$, with $T_s = 1/(1500 \times 2048)$. Two radio frame structures are supported:

- Type 1, applicable to FDD
- Type 2, applicable to TDD

Frame Structure type 1

Frame structure 1 is applicable to both full duplex and half duplex FDD. As shown in Figure 20, Each radio frame has 10 ms long and consists of 20 slots of length 0.5 ms numbered from 0 to 19. A subframe is defined as 2 consecutive slot where subframe i consists of slots 2i and 2i+1. For FDD, 10 subframes are available for downlink transmission and 10 subframes are available for uplink transmissions in each 10 ms interval.

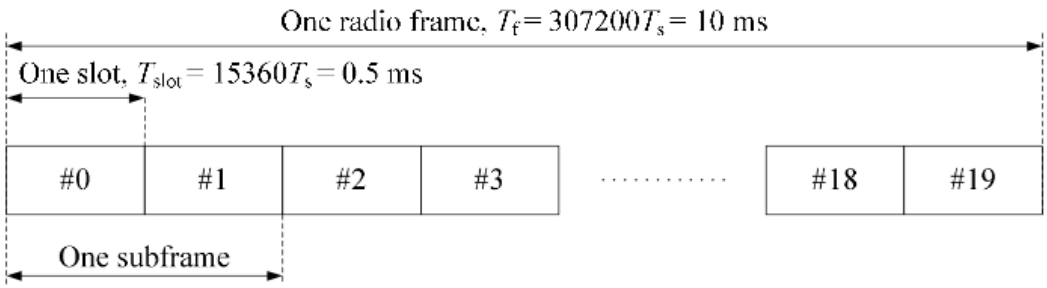


Figure 20. Frame structure type 1

Frame Structure type 2

Frame structure type 2 is applicable to TDD. As we can see in the Figure 21, Each radio frame has 10 ms long and consists of 2 half-frames of length of 5ms each. Each half-frame consists of 5 subframe of length 1ms. A subframe is defined as 2 consecutive slot where subframe i consists of slots 2i and 2i+1. The Figure 22 shows the supported uplink-downlink configurations, where for each subframe in radio frame, 'D' denotes the subframe reserved for downlink transmissions, 'U' subframe reserved for uplink transmission, and 'S' denotes the special subframe with the three fields DwPTS, GP, UpPTS (their value can be seen in Figure 23).

Uplink-downlink configuration with both 5 ms and 10 ms downlink-to-uplink switch-point periodicity are supported. In case of 5 ms downlink-to-uplink switch-point periodicity, the special subframe exists in both half-frame while in case of 10 ms downlink-to-uplink switch-point periodicity, the special subframe exists in the first half-frame only. Subframes 0 and 5 and DwPTS are always reserved for downlink transmission, and the UpPTS and the subframe immediately following the special subframe are always reserved for uplink transmission.

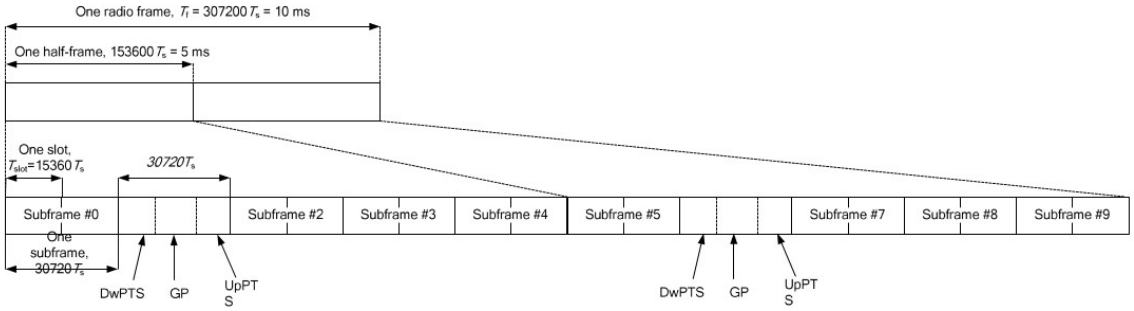


Figure 21. Frame structure type 2(for 5ms switch-point periodicity)

Uplink-downlink configuration	Downlink-to-Uplink Switch-point periodicity	Subframe number									
		0	1	2	3	4	5	6	7	8	9
0	**5 ms**	**D**	**S**	**U**	**U**	**U**	**D**	**S**	**U**	**U**	**U**
1	**5 ms**	**D**	**S**	**U**	**U**	**D**	**D**	**S**	**U**	**U**	**D**
2	**5 ms**	**D**	**S**	**U**	**D**	**D**	**D**	**S**	**U**	**D**	**D**
3	**10 ms**	**D**	**S**	**U**	**U**	**U**	**D**	**D**	**D**	**D**	**D**
4	**10 ms**	**D**	**S**	**U**	**U**	**D**	**D**	**D**	**D**	**D**	**D**
5	**10 ms**	**D**	**S**	**U**	**D**	**D**	**D**	**D**	**D**	**D**	**D**
6	**5 ms**	**D**	**S**	**U**	**U**	**U**	**D**	**S**	**U**	**U**	**D**

Figure 22. Uplink-downlink configurations

Special subframe configuration	Normal cyclic prefix in downlink			Extended cyclic prefix in downlink		
	DwPTS	UpPTS		DwPTS	UpPTS	
		Normal cyclic prefix in uplink	Extended cyclic prefix in uplink		Normal cyclic prefix in uplink	Extended cyclic prefix in uplink
0	$6592 \cdot T_s$	$2192 \cdot T_s$	$2560 \cdot T_s$	$7680 \cdot T_s$	$2192 \cdot T_s$	$2560 \cdot T_s$
1	$19760 \cdot T_s$			$20480 \cdot T_s$		
2	$21952 \cdot T_s$			$23040 \cdot T_s$		
3	$24144 \cdot T_s$			$25600 \cdot T_s$		
4	$26336 \cdot T_s$			$7680 \cdot T_s$		
5	$6592 \cdot T_s$	$4384 \cdot T_s$	$5120 \cdot T_s$	$20480 \cdot T_s$	$4384 \cdot T_s$	$5120 \cdot T_s$
6	$19760 \cdot T_s$			$23040 \cdot T_s$		
7	$21952 \cdot T_s$			-	-	-
8	$24144 \cdot T_s$			-	-	-

Figure 23. Configuration of special subframe

Appendix 3: Root Zadoff-Chu sequence order

In [2], we can find the full version of the table containing all the root ZC sequence order shown in figure 24, 25 and 26.

Logical root sequence number	Physical root sequence number u (in increasing order of the corresponding logical sequence number)
0–23	129, 710, 140, 699, 120, 719, 210, 629, 168, 671, 84, 755, 105, 734, 93, 746, 70, 769, 60, 779 2, 837, 1, 838
24–29	56, 783, 112, 727, 148, 691
30–35	80, 759, 42, 797, 40, 799
36–41	35, 804, 73, 766, 146, 693
42–51	31, 808, 28, 811, 30, 809, 27, 812, 29, 810
52–63	24, 815, 48, 791, 68, 771, 74, 765, 178, 661, 136, 703
64–75	86, 753, 78, 761, 43, 796, 39, 800, 20, 819, 21, 818
76–89	95, 744, 202, 637, 190, 649, 181, 658, 137, 702, 125, 714, 151, 688
90–115	217, 622, 128, 711, 142, 697, 122, 717, 203, 636, 118, 721, 110, 729, 89, 750, 103, 736, 61, 778, 55, 784, 15, 824, 14, 825
116–135	12, 827, 23, 816, 34, 805, 37, 802, 46, 793, 207, 632, 179, 660, 145, 694, 130, 709, 223, 616
136–167	228, 611, 227, 612, 132, 707, 133, 706, 143, 696, 135, 704, 161, 678, 201, 638, 173, 666, 106, 733, 83, 756, 91, 748, 66, 773, 53, 786, 10, 829, 9, 830
168–203	7, 832, 8, 831, 16, 823, 47, 792, 64, 775, 57, 782, 104, 735, 101, 738, 108, 731, 208, 631, 184, 655, 197, 642, 191, 648, 121, 718, 141, 698, 149, 690, 216, 623, 218, 621
204–263	152, 687, 144, 695, 134, 705, 138, 701, 199, 640, 162, 677, 176, 663, 119, 720, 158, 681, 164, 675, 174, 665, 171, 668, 170, 669, 87, 752, 169, 670, 88, 751, 107, 732, 81, 758, 82, 757, 100, 739, 98, 741, 71, 768, 59, 780, 65, 774, 50, 789, 49, 790, 26, 813, 17, 822, 13, 826, 6, 833
264–327	5, 834, 33, 806, 51, 788, 75, 764, 99, 740, 96, 743, 97, 742, 166, 673, 172, 667, 175, 664, 187, 652, 163, 676, 185, 654, 200, 639, 114, 725, 189, 650, 115, 724, 194, 645, 195, 644, 192, 647, 182, 657, 157, 682, 156, 683, 211, 628, 154, 685, 123, 716, 139, 700, 212, 627, 153, 686, 213, 626, 215, 624, 150, 689

Figure 24. Root ZC sequence order for format 0-3. part 1

328–383	225, 614, 224, 615, 221, 618, 220, 619, 127, 712, 147, 692, 124, 715, 193, 646, 205, 634, 206, 633, 116, 723, 160, 679, 186, 653, 167, 672, 79, 760, 85, 754, 77, 762, 92, 747, 58, 781, 62, 777, 69, 770, 54, 785, 36, 803, 32, 807, 25, 814, 18, 821, 11, 828, 4, 835
384–455	3, 836, 19, 820, 22, 817, 41, 798, 38, 801, 44, 795, 52, 787, 45, 794, 63, 776, 67, 772, 72 767, 76, 763, 94, 745, 102, 737, 90, 749, 109, 730, 165, 674, 111, 728, 209, 630, 204, 635, 117, 722, 188, 651, 159, 680, 198, 641, 113, 726, 183, 656, 180, 659, 177, 662, 196, 643, 155, 684, 214, 625, 126, 713, 131, 708, 219, 620, 222, 617, 226, 613
456–513	230, 609, 232, 607, 262, 577, 252, 587, 418, 421, 416, 423, 413, 426, 411, 428, 376, 463, 395, 444, 283, 556, 285, 554, 379, 460, 390, 449, 363, 476, 384, 455, 388, 451, 386, 453, 361, 478, 387, 452, 360, 479, 310, 529, 354, 485, 328, 511, 315, 524, 337, 502, 349, 490, 335, 504, 324, 515
514–561	323, 516, 320, 519, 334, 505, 359, 480, 295, 544, 385, 454, 292, 547, 291, 548, 381, 458, 399, 440, 380, 459, 397, 442, 369, 470, 377, 462, 410, 429, 407, 432, 281, 558, 414, 425, 247, 592, 277, 562, 271, 568, 272, 567, 264, 575, 259, 580
562–629	237, 602, 239, 600, 244, 595, 243, 596, 275, 564, 278, 561, 250, 589, 246, 593, 417, 422, 248, 591, 394, 445, 393, 446, 370, 469, 365, 474, 300, 539, 299, 540, 364, 475, 362, 477, 298, 541, 312, 527, 313, 526, 314, 525, 353, 486, 352, 487, 343, 496, 327, 512, 350, 489, 326, 513, 319, 520, 332, 507, 333, 506, 348, 491, 347, 492, 322, 517
630–659	330, 509, 338, 501, 341, 498, 340, 499, 342, 497, 301, 538, 366, 473, 401, 438, 371, 468, 408, 431, 375, 464, 249, 590, 269, 570, 238, 601, 234, 605
660–707	257, 582, 273, 566, 255, 584, 254, 585, 245, 594, 251, 588, 412, 427, 372, 467, 282, 557, 403, 436, 396, 443, 392, 447, 391, 448, 382, 457, 389, 450, 294, 545, 297, 542, 311, 528, 344, 495, 345, 494, 318, 521, 331, 508, 325, 514, 321, 518
708–729	346, 493, 339, 500, 351, 488, 306, 533, 289, 550, 400, 439, 378, 461, 374, 465, 415, 424, 270, 569, 241, 598
730–751	231, 608, 260, 579, 268, 571, 276, 563, 409, 430, 398, 441, 290, 549, 304, 535, 308, 531, 358, 481, 316, 523

Figure 25. Root ZC sequence order for format 0-3. part 2

Logical root sequence number	Physical root sequence number u (in increasing order of the corresponding logical sequence number)																			
0 – 19	**1**	**138**	**2**	**137**	**3**	**136**	**4**	**135**	**5**	**134**	**6**	**133**	**7**	**132**	**8**	**131**	**9**	**130**	**10**	**129**
20 – 39	**11**	**128**	**12**	**127**	**13**	**126**	**14**	**125**	**15**	**124**	**16**	**123**	**17**	**122**	**18**	**121**	**19**	**120**	**20**	**119**
40 – 59	**21**	**118**	**22**	**117**	**23**	**116**	**24**	**115**	**25**	**114**	**26**	**113**	**27**	**112**	**28**	**111**	**29**	**110**	**30**	**109**
60 – 79	**31**	**108**	**32**	**107**	**33**	**106**	**34**	**105**	**35**	**104**	**36**	**103**	**37**	**102**	**38**	**101**	**39**	**100**	**40**	**99**
80 – 99	**41**	**98**	**42**	**97**	**43**	**96**	**44**	**95**	**45**	**94**	**46**	**93**	**47**	**92**	**48**	**91**	**49**	**90**	**50**	**89**
100 – 119	**51**	**88**	**52**	**87**	**53**	**86**	**54**	**85**	**55**	**84**	**56**	**83**	**57**	**82**	**58**	**81**	**59**	**80**	**60**	**79**
120 – 137	**61**	**78**	**62**	**77**	**63**	**76**	**64**	**75**	**65**	**74**	**66**	**73**	**67**	**72**	**68**	**71**	**69**	**70**	**-**	**-**
138 – 837	**N/A**																			

Figure 26. Root ZC sequence order for format 4

Appendix 4: N_{CS} values

The table shown in the table below can be got in [2].

N_{CS} configuration	N_{CS} value	
	Unrestricted set	Restricted set
0	0	15
1	13	18
2	15	22
3	18	26
4	22	32
5	26	38
6	32	46
7	38	55
8	46	68
9	59	82
10	76	100
11	93	128
12	119	158
13	167	202
14	279	237
15	419	-

Figure 27. N_{CS} for preamble generation (preamble formats 0-3)

N_{CS} configuration	N_{CS} value
0	2
1	4
2	6
3	8
4	10
5	12
6	15
7	N/A
8	N/A
9	N/A
10	N/A
11	N/A
12	N/A
13	N/A
14	N/A
15	N/A

Figure 28. N_{CS} for preamble generation (preamble format 4) Appendix 5: Random access baseband parameters.

Appendix 5: Random access baseband parameters.

Preamble format	Δf_{RA}	φ
0 – 3	1250 Hz	7
4	7500 Hz	2

Figure 29. Random access baseband parameters

Appendix 6: Frame structure type 2 random access configurations for preamble format 0-4.

More information about the frame structure random access configuration can be found in [2].

PRACH configuration Index	Preamble Format	Density Per 10 ms (D_{RA})	Version (r_{RA})	PRACH configuration Index	Preamble Format	Density Per 10 ms (D_{RA})	Version (r_{RA})
0	0	0.5	0	32	2	0.5	2
1	0	0.5	1	33	2	1	0
2	0	0.5	2	34	2	1	1
3	0	1	0	35	2	2	0
4	0	1	1	36	2	3	0
5	0	1	2	37	2	4	0
6	0	2	0	38	2	5	0
7	0	2	1	39	2	6	0
8	0	2	2	40	3	0.5	0
9	0	3	0	41	3	0.5	1
10	0	3	1	42	3	0.5	2
11	0	3	2	43	3	1	0
12	0	4	0	44	3	1	1
13	0	4	1	45	3	2	0
14	0	4	2	46	3	3	0
15	0	5	0	47	3	4	0
16	0	5	1	48	4	0.5	0
17	0	5	2	49	4	0.5	1
18	0	6	0	50	4	0.5	2
19	0	6	1	51	4	1	0
20	1	0.5	0	52	4	1	1
21	1	0.5	1	53	4	2	0
22	1	0.5	2	54	4	3	0
23	1	1	0	55	4	4	0
24	1	1	1	56	4	5	0
25	1	2	0	57	4	6	0
26	1	3	0	58	N/A	N/A	N/A
27	1	4	0	59	N/A	N/A	N/A
28	1	5	0	60	N/A	N/A	N/A
29	1	6	0	61	N/A	N/A	N/A
30	2	0.5	0	62	N/A	N/A	N/A
31	2	0.5	1	63	N/A	N/A	N/A

Figure 30. frame structure type 2 random access configurations

Glossary

A list of all acronyms and the meaning they stand for

3GPP: 3rd Generation Partnership Project

CP: Cyclic Prefix

DL: Downlink

FDD-LTE: Frequency Division Duplexing Long Term Evolution

GT: Guard Time

LTE: Long Term Evolution

OOP: Oriented Object Programming

RACH: Random Access Channel

PAPR: Peak-to-Average Power Ratio

PDP: Power Delay Profile

PRACH: Physical Random Access Channel

RA: random access

TD-LTE: Time Division Duplexing Long Term Evolution

UE: User Equipment

UL: Uplink

SDR: Software Defined Radio