IMPLEMENTATION OF DYNAMIC RESOURCE ALLOCATION FOR LTE SYSTEM USING GNU RADIO

by

Hanke Cheng

A Thesis
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Master of Science
Electrical and Computer Engineering

Committee:

______________________	Dr. Bijan Jabbari, Thesis Director
______________________	Dr. Brian L. Mark, Committee Member
______________________	Dr. Yariv Ephraim, Committee Member
______________________	Dr. Monson Hayes, Department Chair
______________________	Dr. Kenneth S. Ball, Dean, Volgenau School of Engineering
Date: December 3, 2014	Fall Semester 2014 George Mason University Fairfax, VA

Implementation of Dynamic Resource Allocation for LTE System Using GNU Radio

A Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at George Mason University

by

Hanke Cheng
Bachelor of Science
Wuhan University of Technology, 2011

Director: Bijan Jabbari, Professor
Department of Electrical and Computer Engineering

Fall Semester 2014
George Mason University
Fairfax, VA

DEDICATION

This is dedicated to my parents Jishan Cheng and Qin Wang, for their unconditional support and love, and to my girl friend, Yi Zhang.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Bijan Jabbari, who provided all of the academic advices and the devices I need during my work of Master thesis. With his encouragement and stimulation, he helped me find my way to the completion of my research and improved my research ability.

I thank Dr. Mark and Dr. Ephraim for serving on my committee. I thank Dr. Mark for his knowledge of GNU Radio. I also thank Dr. Ephraim for the fundamental course in communication theory, which is helpful for my research.

I thank my friend Pei Li and Hao Sun, who helped me tremendously on programming. I also thank all of my friends in Communications and Networking Lab (CNL), especially Ben for useful discussion about GNU Radio technologies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

IMPLEMENTATION OF DYNAMIC RESOURCE ALLOCATION FOR LTE SYSTEM USING GNU RADIO

Hanke Cheng, M.S.

George Mason University, 2014

Thesis Director: Bijan Jabbari

Nowadays, the fourth generation mobile communication is being popularized all over the world. As the core technologies, OFDM and MIMO are being widely researched and attracting more and more people to make them perfect. Dynamic resource allocation for OFDM is one of the research topics in the field of OFDM technology for Long Term Evolution (LTE) standard. In order to dynamically allocate resource to multiple users with different purposes, there is large number of algorithms used in specific situations. On the other hand, a universal technology named Software Defined Radio (SDR) is rapidly developing for signal processing and wireless communication. With more flexibility and lower cost, SDR, especially GNU Radio, are widely used for researching and building portable communication devices. This thesis combines LTE and GNU Radio technology.

This thesis consists of two parts: one is a study of the important issues in wireless communication and dynamic resource allocation for OFDM based systems; the other is the implementation of a simplified signal processing structure for an LTE communication system using GNU Radio. Within this signal processing structure, the dynamic resource allocation algorithms are embedded in the OFDM modulator.

In this thesis, we choose three typical dynamic resource allocation algorithms: Max-Sum, Max-Min and Nash Bargaining Game. We assume a general LTE network with one eNodeB base station and multiple user devices. Max-Sum algorithm maximizes the total throughput of the network since it only allocates channels to those users who have the best channel gain. Max-Min algorithm maximizes the worst user device rate to guarantee that the user device which is in the worst channel environment can obtain an opportunity to access to the network and maintain an acceptable quality of service. Nash bargaining game algorithm maximizes the product of the rates of each user device. This algorithm provides an approach to enlarge the total throughput and guarantees the fairness of the occupancy for each user devices.

Using GNU Radio to implement dynamic resource allocation requires a large amount of prerequisites. GNU Radio, a kind of Software Defined Radio framework, provides a C++ library for signal processing. It provides GNU Radio Companion (GRC) for researchers to build a signal processing flow graph for a specific task. GRC translates the flow graph to Python codes and the Python script calls the compiled C++ programs which process the signals in encapsulated blocks. Based on the rules of GNU Radio development and some built-in signal processing modules, we build the OFDM modulator and demodulator for downlink communication and the SC-FDMA modulator and demodulator for uplink communication, which are the technologies adopted by the LTE standard. More important, we design and build the base station and user device to simulate real situations in which people use mobile phones to send requests and get information. Then, the three dynamic resource allocation algorithms are embedded in the resource allocation module of the base station to study the differences among them.

CHAPTER 1: INTRODUCTION

With the growing requirement of better quality of service (QoS) and faster rate of data transmission, the gap between two generations of wireless communication system becomes smaller. The fourth generation wireless communication system has just been popularized recently. However, the fifth generation wireless communication standard is being prepared for drafting. As we know, each new generation develops new communication technologies and brings a tremendous improvement of wireless communication.

In order to promote the rate of data transmission and better quality of service, wide band technologies are used in the wireless communication system. However, the major challenges for wide band communication are the limited spectrum resource and the frequency-selective fading caused by multipath transmission. As a result, more efficient utilization of spectrum resource and better performance of overcoming nature of the wireless channel are required to lead a revolution of wireless communication. Orthogonal Frequency Division Multiplexing (OFDM) is the leading technology which overcomes those problems. The fourth generation standard, Long Term Evolution (LTE), adopts it as the downlink modulation. Besides, some Wireless Local Area Network also applies OFDM as the primary modulation. The well-known standard before the birth of LTE, WiMAX, is also using OFDM modulation to approach to a higher rate. The idea of

OFDM modulation is to divide a wide bandwidth into small pieces that the impulse response of the small bands overlaps with each other. However, all of the small bands are orthogonal to each other so that there is no interference between the adjacent bands but the utilization of spectrum resource is maximized. Since OFDM modulation is to divide a wide band into a large number of sub-channels with narrow band, the bandwidth of one channel usually smaller than the coherence bandwidth of frequency selective fading channel. As a result, the fading on the channel is flat most of the time.

When using OFDM modulation in wireless communication system, the resource allocation becomes the critical aspect of the effect on the performance. If there is only one user, all of the subcarriers are occupied by this user. While in multi user system, the algorithm of resource allocation affects the equipment cost, throughput, capacity, fairness and so forth. The fixed resource allocation has the lowest complexity of the equipment but the efficiency is also low since it has to face the varying channel environment. In order to improve the efficiency, dynamic resource allocation is introduced and applied in most of the wireless communication systems. The dynamic resource allocation algorithms vary for specific systems and devices so that it requires new equipments for different kind of use. To improve the flexibility, universal technologies are usually better solutions.

As a kind of universal technology, Software Defined Radio technology requires fewer types of equipment but does a variety of tasks. As mentioned above, different dynamic resource allocation algorithms require different equipments. However, if using SDR technology, the number of required devices can probably be limited to one or two.

This is due to the mechanism of SDR, which uses software to process signal via universal equipment, such as personal computer. As an outstanding framework of SDR, GNU Radio only needs a personal computer and a Universal Software Radio Peripheral (USRP). The computer processes the signal and USRP transmits and receives signal through the air. With the high flexibility and low cost, the GNU Radio platform lowers the threshold of real world based research. This fact contributes to more useful experiment with the theoretical research in the field of wireless communication and signal process.

1.1 Statement of the Problem

A large number of researches on dynamic resource allocation are done with the development of OFDM based communication system. However, most of the researches are theoretical work with some analysis done by the simulation software such as MATLAB. In reality, if the theoretical research introduces an infeasible technology for real life use, the research loses its value.

As the development of GNU Radio technology, a low cost real world experiment platform is possible for every laboratory conducting the research in the field of wireless communication. Regard as a popular research field, dynamic resource allocation and other practical researches need to move to SDR platform such as GNU Radio in order to achieve more valuable result from real world experiments.

1.2 Related Works

1.2.1 Dynamic Resource Allocation in OFDM Based System

The problems of dynamic resource allocation in OFDM based system are widely researched in [6, 7, 10, 11, 14]. Some algorithms for optimizing the dynamic resource allocation are put forward, as well as their modifications for specific purpose. In [6], the algorithm maximizes the rate of the user which has the worst channel condition to guarantee the fairness for all of the users. However, it significantly drops the total rate of the system since the sacrifice of the users with better channel condition. In contrary, [7] puts forward an algorithm that maximizes the total rate of the system by allocating resources only to the user which has best channel gain. As a result, the total rate is improved but the worst user may face high latency, and even lose the connection. In order to guarantee the fairness and obtain higher rate at the same time, an algorithm using Nash Bargaining Solution is developed in [14]. Except for these basic algorithms, there are a number of algorithms similar to them and deal with different situations [8, 10, 12, 13].

1.2.2 Software Defined Radio

With the fast growing wireless communication network, modern people get more opportunities to achieve high speed network service. At the same time, the improved design and fabrication of the chips and devices of communication system become less expensive. However, every component in different communication systems is customized which is solidified in specified system. In order to overcome the low flexibility, software defined radio was born and GNU Radio became the most outstanding one in the open source world.

Decades ago, the performance of personal computer was extremely restricted because of the limited scale and fabrication of chips. CPU, for instance, is responsible for all of the commands and calculation. As a result, the low upper limit of CPU allows it to take small burden work. As the fast development of CPU and other chips, heavier burden of work is able to be processed by personal computer. Thus, it is possible to process signals in communication systems so that software defined radio such as GNU radio is being developed in recent years.

GNU Radio is a framework for signal processing via small computers. Instead of a solid chip, developing programs to process signal makes system more flexible and lower cost. By using some algorithms, every personal computer can process digital signal in the same way defined by specific programs. The flexibility, reusability and universality make GNU Radio known as the revolution of wireless communication. Take OpenBTS as the typical example, it follows the second generation (2G) mobile communication standard and implements the entire functions of mobile communication system. With Universal Software Radio Peripheral (USRP) designed by Matt Ettus, the personal computer running OpenBTS turns into a base station of 2G mobile communication. Nowadays, the fourth generation (4G) mobile communication technology, LTE, does achieve milestones. Amerisoft LTE published some products which implements eNodeB, the base station of 4G mobile communication, via USRP. Another group, OpenLTE, is working hard to provide a free LTE module of GNU Radio. Regard as those works, the low cost and portable 4G base station will promote the coverage of 4G mobile communication network.

Not only contribute to mobile communication system, GNU Radio has much deeply extensibility for other field of radio systems. By utilizing available modules of GNU Radio, it can be developed as an FM radio, a surveillance system in the field of aerospace, and so forth. GNU Radio fulfills the requirement of any technologies of digital signal processing since all the processing is able to be done by proper algorithms.

1.3 Structure of the Thesis

This thesis is more practical because the goal is to build the LTE uplink and downlink to establish the environment for dynamic resource allocation research on GNU Radio platform. Chapter 2 introduces the details of main aspects of dynamic resource allocation. Three typical algorithms, Max-Min, Max-Sum and Nash Bargaining Game, are introduced in this chapter. Chapter 3 explains the mechanism of GNU Radio framework for easier understanding of the work with GNU Radio. Chapter 4 describes the communication logic between base station and user device, which is designed for simple implementation of the downlink and uplink technologies in LTE system by GNU Radio. The full implementation of downlink and uplink is elaborated. Chapter 5 presents an experiment of the environment using the three kinds of dynamic allocation algorithms. Conclusions and future work are discussed in Chapter 6.

CHAPTER 2: DYNAMIC RESOURCE ALLOCATION

This thesis focuses on dynamic resource allocation algorithm for OFDM based wireless communication systems, especially LTE system. We will explain the water-filling algorithm for single user dynamic resource allocation and three algorithms, Max-Sum, Max-Min and Max-Product, for multi-user dynamic resource allocation. All of these algorithms represent the original work on their own directions and a large number of optimized algorithms for specific environments have been developed based on these classic algorithms. This chapter discusses the major issues of wireless channel characteristics first, then the dynamic resource allocation algorithms.

2.1 Wireless Channel Characteristics

Unlike the wired channel, wireless channel is not constrained in a solid medium. This fact contributes to a harsh channel environment because of these three phenomenons: multi-path transmission, shadowing and time varying characteristics.

2.1.1 Multi-path

A typical multi-path transmission situation is shown in Figure 2.1. The triangles stand for the transmitters or receivers while the rectangles represents obstacles such as constructions and vegetations. The transmission path 2 is the signal directly transmitted from the antenna of the transmitter to that of the receiver. The transmission path 1,3 and 4

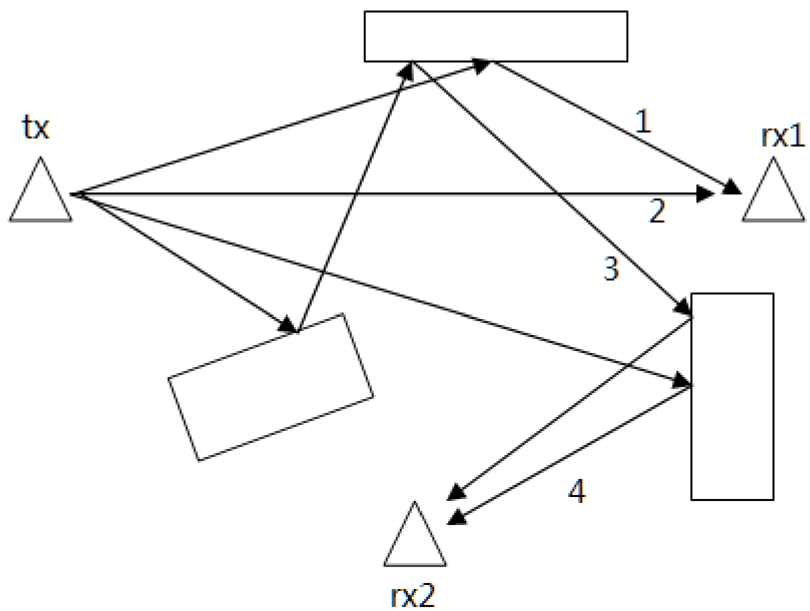
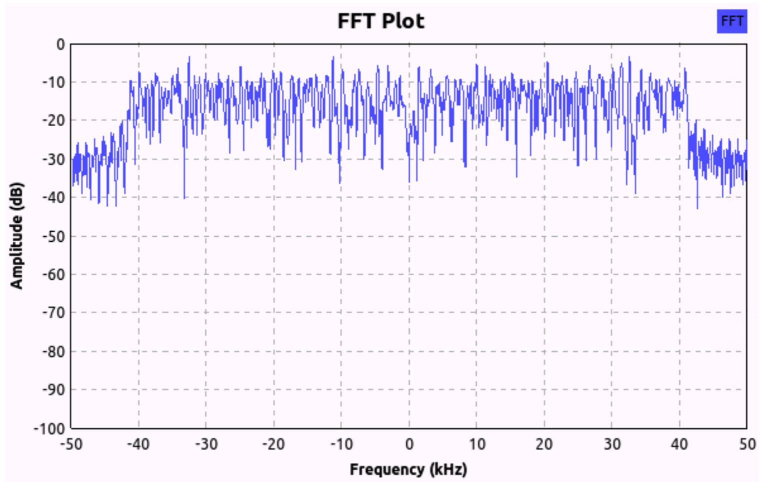


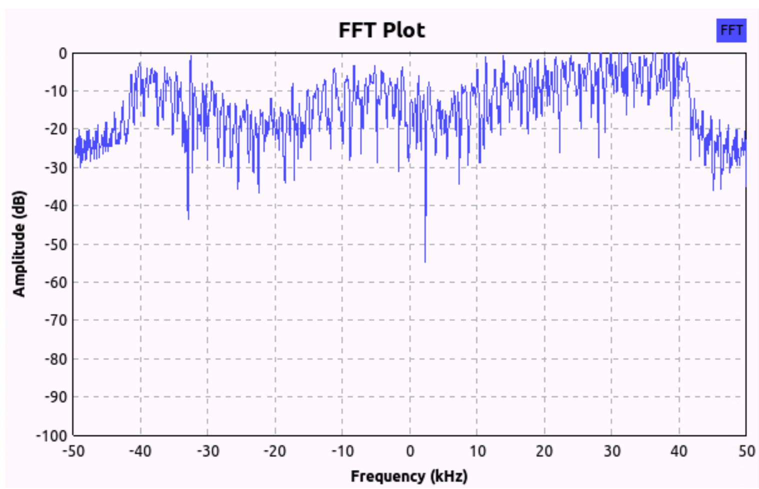
Figure 2. 1 An example for multi-path transmission and shadowing

include reflection when the transmitter sends signal to the receiver, which leads to multi-path interference effecting the direct transmission signal. As the matter of fact, the extra transmission path exists numerously in the wireless channel which contributes to significant fading of the signal at the receiver.

Multi-path fading can be modeled by a Rayleigh distribution. In frequency domain the signal shows a significant amplitude drop on some frequencies so that the multi-path fading is also called frequency selective fading, which can be classified as fast fading. Figure 2.2a is the OFDM signal in frequency domain only with Additive White Gaussian Noise (AWGN) and Figure 2.2b applies the frequency selective fading on the



(a) The signal transmitted only through AWGN channel



(b) The signal transmitted only through frequency selective fading channel

Figure 2. 2 The comparison between the signals transmitted through AWGN channel and frequency selective fading channel

signal. Comparing the FFT plot of the same signal, the amplitude of the signal with only AWGN channel shows a relatively even throughout the bandwidth, while the frequency selective fading leads to amplitude drops in the range from -30 to -20 kHz and from 0 to 10 kHz. To overcome the fast fading channel, the bandwidth of the channel has to smaller than the coherence bandwidth so that the channel avoids going across the amplitude dropping area. Since the bandwidth of each channel of OFDM modulation is narrow, OFDM modulation has better performance in frequency selective fading channels.

2.1.2 Shadowing

Shadowing is caused by the situation that the signal cannot be transmitted to the receiver directly. In other word, if the receiver is under the shadow of the transmitter, the signal can only reach the receiver by reflection from other ways. Hence, shadowing completely depends on multi-path transmission. The Figure 2.1 shows the situation of shadowing. The direct transmission path from the transmitter and the second receiver is blocked by a obstacle so that the signal need to find other paths such as path 3 and 4. Shadowing is model as lognormal distribution which implies that the fading is categorized as slow fading. The impact on the amplitude of the signal is much gentler over the entire bandwidth than frequency selective fading.

2.1.3 Time Varying

Time varying characteristics is due to the mobility of receiver relatively to transmitter. When the relative speed of receiver and transmitter is larger than 0, the Doppler frequency shift occurs. If the receiver is approaching to the transmitter, the central frequency of the received signal shifts positively comparing to the transmitted signal and vice versa. Figure 2.3 shows the OFDM signal with positive frequency shift. Compare to Figure 2.2a, the central frequency of the signal shifts approximately 6 kHz. In order to correct the frequency offset, an appropriate frequency offset estimation

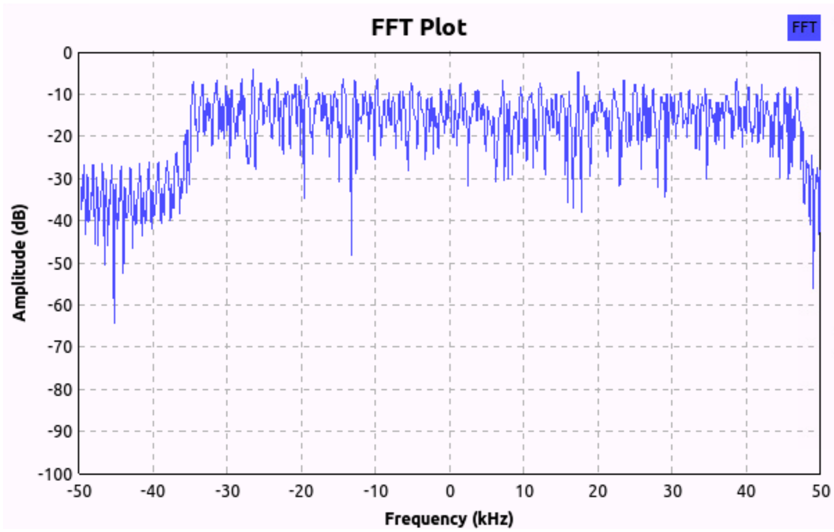


Figure 2. 3 The received signal with positive Doppler shift

algorithm is essential for signal demodulation. Schmidl and Cox algorithm [4] is a typical one implemented in OFDM based communication system.

2.2 Resource Allocation for Single User

Assume that there is only one user device communicating with base station, all of the OFDM subcarriers are allocated to the user device. Due to the frequency selective fading, some of the subcarriers are affected by severe channel environment. In this case, the probability of packet loss increases and the rate decreases if the system still allocates these subcarriers to the user device. To overcome the rate loss and maximized the throughput, water-filling algorithm is introduce in the single user dynamic resource allocation [22].

According to Shannon's Equation, we can obtain the maximum rate for the single user device as

$$R = \sum_{n=1}^{N} B \log_2 (1 + \frac{P_n h_n^2}{\sigma^2}) \tag{2.1}$$

where B is the bandwidth, N is the number of subcarriers, Pn is the power on n-th subcarrier and h_n is the channel envelope. If we apply the method of Lagrange multiplier to maximize R, we obtain

$$P_n + \frac{\sigma^2}{h_n{}^2} = \frac{B}{\lambda} \tag{2.2}$$

where λ is the Lagrange multiplier and $\sigma^2 / h_n{}^2$ stands for Noise-to-Carrier Ratio (NCR). For all of the subcarriers, we have

$$\frac{1}{\lambda} = \frac{P + \sum_{n=1}^{N} \sigma^2 / h_n{}^2}{NB} \tag{2.3}$$

where P is the maximum total power of transmitted signal.

Water-filling algorithm iteratively calculates the amount of power that allocates to every subcarrier based on the total provided power and the carrier-to-noise ratio of every subcarrier until no negative amount of power is figured out in the calculation. The carrier-to-noise ratio of every subcarrier is obtained from the channel. The negative amount of power is contributed by low carrier-to-noise ratio, and the algorithm does not allocate power to the subcarriers with such a low carrier-to-noise ratio. After calculation, the sum of the allocated power and NCR on every subcarrier, except for the subcarriers with no power allocated, is a constant value. The constant $1/\lambda$ in equation 2.3 is the constant value, so called water level.

Figure 2.4 shows the result of an example implementing water-filling algorithm. The total amount of power is 0.1 mW allocated to 16 subcarriers. The bandwidth is 1

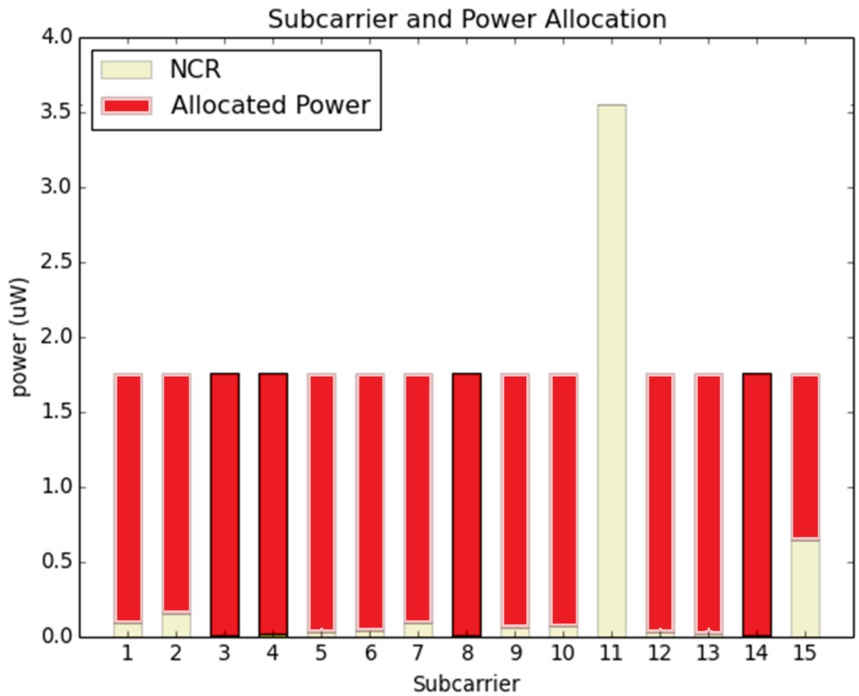


Figure 2. 4 Water filling algorithm for single user

MHz and the noise density is -80 dBm. The power allocated to each subcarrier is marked as red rectangles and the NCR on each subcarrier is marked as yellow rectangles beneath the red rectangles. We can see subcarrier with the NCR higher than the water level will be left empty.

2.3 Resource Allocation for Multi-user

Although the single user dynamic resource allocation can be dealt with water-filling algorithm, the wireless communication system usually supports more than one user devices which have distinct channel environment. To optimize the resource allocation, some algorithms for different purpose are developed. This section introduces three typical multi-user dynamic resource allocation algorithms.

Table 2. 1 Subcarrier allocation by Max-Sum algorithm

1.	Initialization: i. Set $R_i = 0$ for all $i = 1, 2, ..., \mathrm{K}$. ii. Set $S = \{1, 2, ..., N\}$.
2.	While $S \neq \emptyset$ i. Find (i, a) satisfying $\lvert G_{i,a} \rvert \geq G_{i,j}$ for all $j \in S$. ii. Update R_i and S for a: $R_i = R_i + C(G_{i,a})$ and $S = S - \{a\}$

Table 2. 2 Subcarrier allocation by Max-Min algorithm

1.	Initialization: i. Set $R_i = 0$ for all $i = 1, 2, ..., \mathrm{K}$. ii. Set $S = \{1, 2, ..., N\}$.
2.	For i = 1 to K, i. Find a satisfying $\lvert G_{i,a} \rvert \geq G_{i,j}$ for all $j \in S$. ii. Update R_i and S for a: $R_i = C(G_{i,a})$ and $S = S - \{a\}$
3.	While $S \neq \emptyset$ i. Find i satisfying $R_i \leq R_n$ for all $n \in [1, \mathrm{K}]$ ii. For the n corresponding to found i, Find a satisfying $\lvert G_{i,a} \rvert \geq G_{i,j}$ for all $j \in S$. iii. Update R_i and S for a: $R_i = R_i + C(G_{i,a})$ and $S = S - \{a\}$

2.3.1 Max-Sum

Max-Sum algorithm maximizes the total rate through finding the subcarriers with maximum channel gain [7]. This algorithm has low complexity since the only work is to find the largest channel gain on each subcarrier for all of the users. The detail of the algorithm is shown in Table 2.1. R_i stands for the current rate of i-th user when allocating subcarrier. The total number of users is K. S stands for the set of subcarriers. $G_{i,a} = h_{i,a}{}^2 / \sigma^2$ and $C(G_{i,a})$ is the Shannon's equation for calculating the maximum rate based on the Channel Gain to Noise Ratio (CGNR).

Table 2. 3 Subcarrier allocation by Max-Product algorithm

1. Initialization: i. Set $R_i = 0$ and $\Omega_i = \varnothing$ for all $i = 1, 2, ..., \mathrm{K}$. ii. Set $S = \{1, 2, ..., N\}$ and $U = \{1, 2, ..., K\}$.
2. While $U \neq \varnothing$, i. Find (i, a) satisfying $\lvert G_{i,a} \rvert \geq G_{i,j}$ for all $i \in U$ and $j \in S$. ii. For the found i , If $R_i \leq R_{i,\min}$,then $\Omega_i = \Omega_i \bigcup \{a\}$, $S = S - \{a\}$, $R_i = R_i + C(G_{i,a})$ else $B = B - \{i\}$
3. While $S \neq \varnothing$, i. Find i satisfying $\frac{G_{i,\mathrm{a}}}{\sum_{j \in \Omega_i} G_{i,j}} \geq \frac{G_{\mathrm{m,a}}}{\sum_{j \in \Omega_m} G_{\mathrm{m},j}}$ for all $m \in [1, \mathrm{K}]$ and $a \in S$ ii. For the found i , Find a satisfying $\lvert G_{i,a} \rvert \geq G_{i,k}$ for all $k \in S$. iii. Update R_i , S and Ω_i for found a : $R_i = R_i + C(G_{i,a})$, $S = S - \{a\}$ and $\Omega_i = \Omega_i \bigcup \{a\}$
4. Power allocation: water fill power for each Ω_i

2.3.2 Max-Min

In order to reduce the probability of connection lost, some researches introduce Max-Min algorithm. It maximizes the capacity of the users which have the worst channel environment [6]. First, the algorithm finds the largest channel gain for each user and allocates corresponding subcarrier to it. At the same time, calculate and store the current rate R for each user. Thus, the algorithm guarantees the opportunity for every user. Second, find the user with the smallest current rate R, then find the second largest channel gain which is on the available subcarrier for this user and allocate corresponding subcarrier to it. Meanwhile, accumulate the current rate R for this user. The second step runs over and over again until all of the subcarriers are occupied. The detail of the algorithm is shown in Table 2.2.

2.3.3 Max-Product

Max-Min algorithm improves the fairness for the users with the worst channel gains. However, maximizing the rate of the user with the worst channel gains sacrifices the rate of users with good channel condition too much. In this case, the total rate of the system keeps at a relatively low level. To optimize the performance, Max-Product algorithm, using Nash Bargaining Solution, is put forward [14]. Similar to Max-min algorithm, the first step is to find the largest channel gain for each user and allocate corresponding subcarrier to it. However, the algorithm sets a minimum rate for each user. If the current user rate is smaller than the minimum rate, the algorithm will continue searching second largest channel gain and allocate one more subcarrier to the user until reaching the minimum rate. After all users reach the minimum rate, the allocation logic moves to the second part. To allocate the rest of subcarriers, the algorithm searches the user which satisfies

$$\frac{G_{i,n}}{\sum_{j\in\Omega_i} G_{i,j}} \geq \frac{G_{\mathrm{m},n}}{\sum_{j\in\Omega_m} G_{\mathrm{m},j}} \qquad (2.4)$$

for all m that 1<m<K where K is the last user. In addition, $G_{i,n}$ is the Channel Gain to Noise Ratio (CGNR) of i-th user on n-th subcarrier and Ω_i is the set of allocated subcarriers of i-th user. This step of the algorithm finds the user on each available subcarrier satisfies that the division of CGNR and the sum of CGNRs on the allocated

subcarriers of the user is larger than the result from the same computation of each other users. If found the user, the algorithm finds the second largest channel gain and allocate the corresponding subcarrier to this user. This step iterates until all of the subcarriers are occupied. In conclusion, Max-Product algorithm takes both rate and fairness into consideration. The detail of the algorithm is shown in Table 2.3. B stands for the set of users. Ω_i represents the set of allocated subcarriers of the i-th user.

CHAPTER 3: GNU RADIO MECHANISM

GNU Radio framework consists of flow graph, GNU Radio Companion (GRC) and signal processing module. Flow graph provides a visualization solution for developers building customized system. GRC translates and executes the system flow graph. Signal processing module is in charge of the work of specific signal processing. In this chapter, we introduce the three components in detail with the example of dial tone, the hello world system in GNU Radio.

3.1 Flow Graph

In order to simplify the work of building system, GNU Radio project developed a visualized tool named GNU Radio Companion (GRC). GRC component is classified in two, signal processing blocks and assistant blocks.

First of all, signal processing blocks take full burden of the work in a system. Each block definitely includes at least one input/output port for digital signal stream transfer. Generally, source blocks only contains output port, sink blocks only contain input block and other signal processing blocks acting as middle operation block contain both input and output block.

Second, assistant blocks do not process any signal. Instead, they make definitions for some variables and create Graphical User Interfaces (GUI) for convenience. Variable block defines an ID as a specific value. If the value changes, the arguments set as the

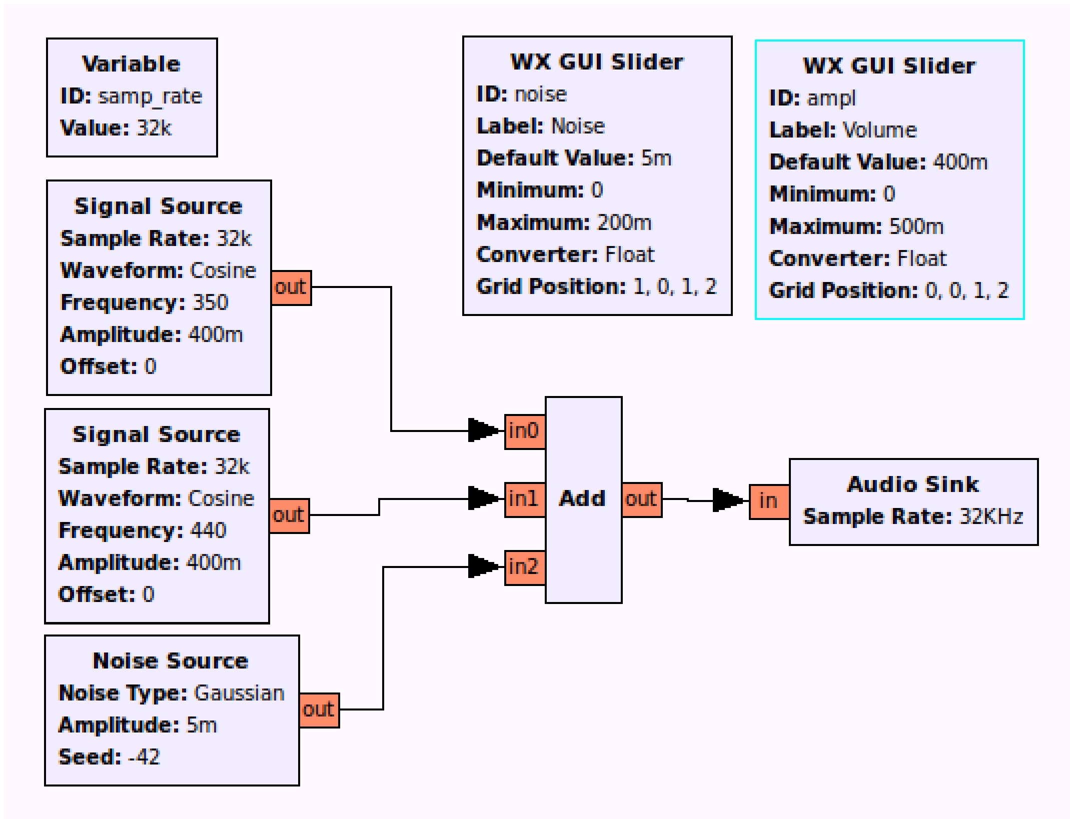
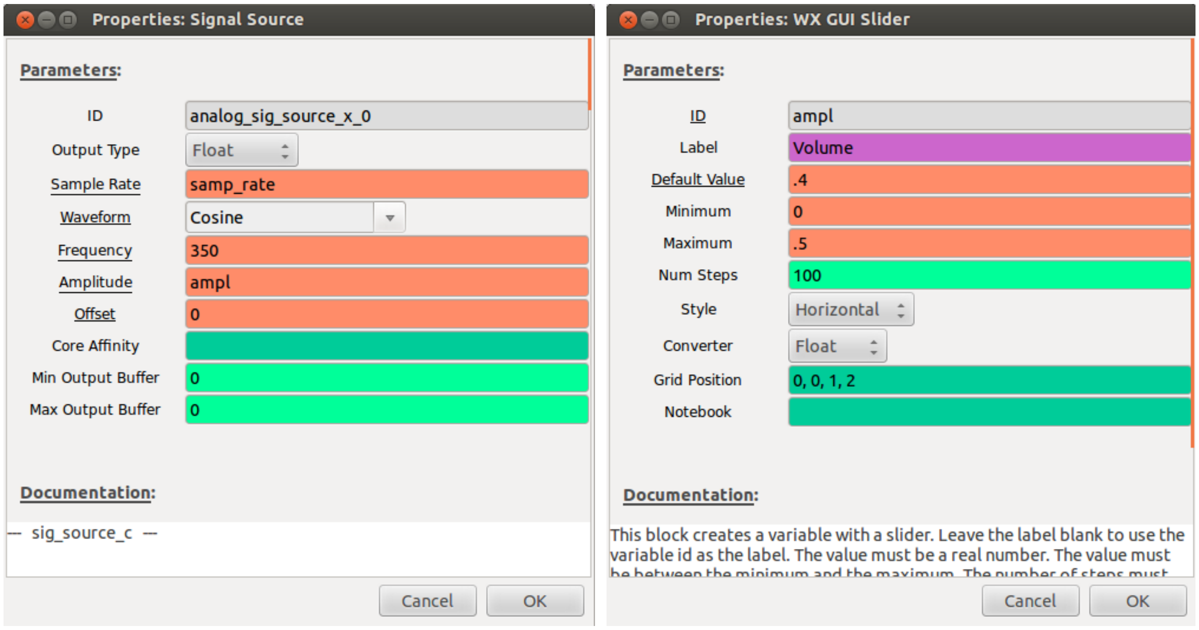


Figure 3. 1 Flow Graph of Audio Signal

corresponding ID in any signal processing blocks changes automatically. In a word, variable block improves the maintenance of systems, especially in sophisticated systems. Another kind of assistant block, GUI block, defines the components for dynamic control of arguments, such as WX GUI slider which controls some values smoothly.

Figure 3.1 shows a typical flow graph of a audio output system. The main part of the system contains 5 components, 2 signal sources, one noise source, one adder and one audio sink, assisted by 2 WX GUI sliders and one variable definition.



(a) Signal Source Property (b) WX GUI Slider Property

Figure 3. 2 Properties of Signal Source and WX GUI Slider Block

Since the system simulates the dial tone when pick up the receiver of phones, the two signal sources generate the sounds with two different frequencies. As the properties shown in the signal source blocks, one sound is generated with frequency 350Hz and the other one with frequency 440Hz. Both signals are in a cosine waveform. In addition, the noise source generates Gaussian noise by pseudo random number for simply simulating the channel noise.

To process these signals, an adder collects all outputs of the three source blocks and adds them together. Finally, the mixed signal is sent to an audio sink, which visits the audio driver and transfer the signal to speakers.

The variable block and GUI blocks improve the test environment. If set the sample rate of signal source block as the ID of variable block, shown as Figure 3.2a, the change of the value of variable block is also applied to the value of sample rate in signal source block. The amplitude property is referred to the value of GUI block shown in Figure 3.2b. Act as a slider, this block defines the minimum and maximum value of the slider, as well as the amount of value for each step. Launching the system, two sounds with different frequency can be heard from the speaker and a dialogue appears for volume and noise adjustment. When moving the slider, the change of the volume of sound and noise occurs distinctly. In conclusion, with the variable block and GUI block, better controllability and flexibility of system simulation are available for research.

3.2 GRC Mechanism

The language that implements the flow graph in GRC is Python. As a script language, Python plays a role of connecting different blocks. Primarily, the object-oriented characteristics of Python simplify the work of building a system via required signal processing modules and blocks. Secondarily, Python is designed as a much advanced programming language in order to minimize the quantity of source code and reduce the burden on programmers. With these advantages, Python is chosen for the critical part of GNU Radio.

GRC translator takes responsibility of converting flow graph into Python program. Since GNU Radio provides a bunch of application program interfaces (API) to connect each signal processing blocks and to build the convenient simulation environment by variable and GUI blocks, writing a Python program to perform the flow graph is not a high burden work. However, GRC releases the repeated work of programming and turns it into establishment of a visualized system flow graph. When creating, configuring and connecting blocks, GRC translator converts them into specific Python codes automatically. The four essential elements of GRC mechanism are as follows.

3.2.1 Block Definition

Sometimes, a block is installed into GNU Radio but not provided by GRC. In this case, the block is available if writing Python code manually while it cannot be implemented in GRC. To solve this problem, the block definition in XML file is introduced by GNU Radio. The frequently used tags in XML file are as follows. Figure 3.3 is the parameter dialog of noise source block in the system shown in Figure 3.1. The ID is a default parameter defined by GNU Radio which represents the name of the new instance of noise source block. This parameter does not influence the operation of the system, only distinguishes the names of the instances defined by the same block.

Table 3.1 The list of frequently used tags in XML block definition file

Block	The envelope of the block definition
Name	The name of the block or a property in this block
Key	The name that be identified by program, such as XML file, required block or data type
Import	The library that is necessary for the block
Make	The name and arguments of the block that will be translated into Python from GRC
Param	The definition of a parameter in the block
Type	The data type of a parameter
Option	The options listed in a parameter
Source	The definition of an output port
Sink	The definition of an input port

The output type defines the type of digital signal stream generated by computer and sent from the output port, such as complex, float, integer and short. The definition in XML file is given in the first “param” tag. In this parameter, four option tags are listed and their names and keys are defined, standing for the four output data types. Since there are four C++ classes written for only one block, the GRC translator needs to know which class is specified. The solution is to use the content of key (type) and opt (fcn) tag corresponding

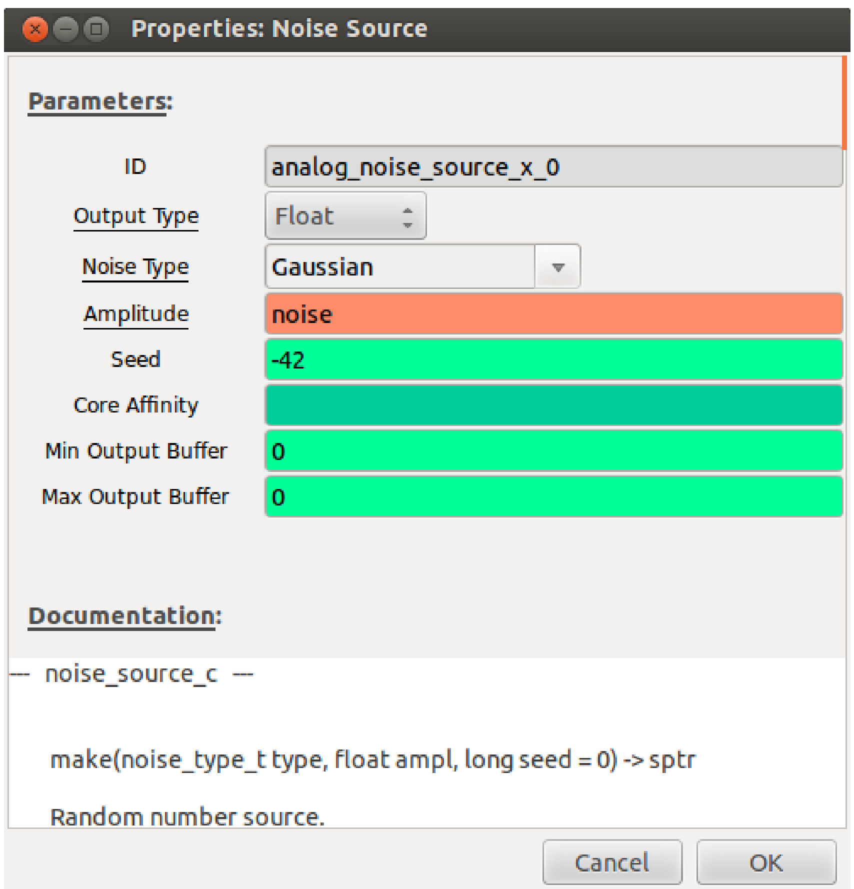


Figure 3. 3 Parameters of Noise Source Block

to them in the content of make tag (type.fcn behind the first dollar sign). When float is chosen, the GRC translator will instantiate *noise_source_f* class with those arguments. Similarly, the noise type parameter contains four options which generate noise signal with difference distribution such as Gaussian and Uniform distribution. The other two parameters, amplitude and seed, are simply defined by the name, key, value and data type. Finally, the source tag defines the output port of the block. Different output data types are represented as varieties of colors. For example, the float output data type is orange (see Figure 3.1).

3.2.2 Block Instantiation and Configuration

Block instantiation and configurations consist of the foundation of GNU Radio signal processing system. Since signal processing blocks are written by C++, Python desires a tool to connect them. For GNU Radio, Simplified Wrapper and Interface Generator (SWIG) is chosen for the responsibility. Based on the compiled C++ files, SWIG generates the interfaces for calling the methods written in C++ by Python programs. After importing related libraries, block establishment is completed by the instantiation of corresponding classes in the libraries and appropriate arguments settings. If using GRC to initialize blocks, simply find the required blocks and drag them to the work space. Then set the properties for each blocks. For example, the first signal source in Figure 3.1 is initialized with sample rate as 32kHz, waveform as cosine, frequency 350Hz, amplitude as 400mV and offset as 0. Translated to Python, it instantiates *sig_source_f* class from library *analog* with the same arguments. Other blocks, including variable and GUI blocks, are instantiated one by one.

3.2.3 Block Connection

After definition and configuration of blocks, a vital method, *connect*, plays a role in the connection of each block in the system. Connection method defines a regular way to establish the association between blocks. First of all, a connection method contains two arguments. Both of them are a pair of instance of block and its port number. Second, the port number of left block represents the output port and that of right block is the input port number. Finally, the specific output port of left block is connected to the specific input port of right block. As a result, the right block can receive the message from left block, which completes the communication of two blocks. In GRC, the connection is established by an arrow between the output port of one block and the input port of the other block. For instance, the output ports of the three signal source are connected to the adder block via three arrows. When starting the simulation, GNU Radio translates the arrows into three *connect* methods with every blocks and ports as their arguments.

3.2.4 Top Block

As the backbone of GRC, the top block is the father of each flow graph. The Python code generated by the flow graph shown in Figure 3.1 proves that every flow graph defined by different systems inherits from top block. The Python code in Appendix 1 shows that the class is inherited from *top_block_gui* in *grc_wxgui*. However, this class is also a child class of top block class. Top block class defines the most important methods for the operation of system. For example, start method, stop method and wait method are the keys to run, stop and keep the system. In addition, as mentioned above, connection method integrates each block into a system. Thus, top block contributes to the real implementation of the entire system.

3.3 Signal Processing Block

Signal processing blocks from the core of GNU Radio. For example, an adder block adds all signals to one output; a signal source generates different kinds of waveforms; BPSK modulator block implements BPSK modulation on input digital signal. Hundreds of signal processing blocks are developed for GNU Radio so that varieties of communication systems are able to be built for researches or as products.

A signal processing block is written by C++ programs. GNU Radio developed a set of C++ APIs for constructing signal processing blocks. The essential parts of a signal processing block are introduced as follows. Take the adder block shown in Figure 3.1 as example.

3.3.1 Smart Pointer

Smart pointer is mainly developed for releasing the burden of manipulating memory when making sophisticated C++ programs. Shared pointer is one of the smart pointers which allow each reference to share the management of the same object. For memory allocation, shared pointer introduces a reference counter to make determination to free the memory or not. If there is one reference points to an object, the reference counter stores 1; if more references share the same object, the value of the reference counter will be the number of the references. When destroy a reference, the reference counter deducts 1. However, the memory occupied by the object will not be freed until the value of the reference counter becomes 0.

3.3.2 IO Signature

IO signature is one of the most important concepts in GNU Radio. As the signal processing block requires input and output, the definition of the input and output ports are the primary work. IO signature requires three arguments, the minimum number of port, the maximum number of port and the size of each port. For instance, in the constructor of *add_ff_impl* class, an input IO signature and an output IO signature are created for the block. For input IO signature, at least one port is required and no upper limit of the number of input port because the second argument is set as -1, while the output IO signature describes that only one output is allowed. Both of the size of input port and output port are the size of float number multiplying by the number of input port. With the IO signature, it guarantees the digital signal streams to be appropriate for the mechanism of this signal processing block, otherwise, the processing will be in chaos.

3.3.3 Type of Block

Typically, signal processing blocks are classified into four types: synchronized block, decimation block, interpolation block and general block.

- Synchronized block guarantees the size of input and output streams to be equal. If the number of digits of input stream for one input port is 1, the output for one port will also produce 1 digit as the output stream.
- Decimation block guarantees the size of input and output streams to be proportional. When the multiple is N, if the number of digits of input stream for one input port is N, the processing will narrow down the signal and the output for one port will produce 1 digit as the output stream.
- Interpolation block is the inverse of decimation block. In a word, 1 digit of input stream will generate N digits of output stream.

- General block does not restrict the size of input and output streams. If the size of input and output streams are not disciplined, general block is the best choice.

The adder block in Appendix 3 inherits from synchronized block. The name of the class of the block and IO signatures are regarded as the arguments when constructing the synchronized block. As a result, the adder block is capable to work as a synchronized block and follow its principles.

3.3.4 Core of Block

The real implementation of signal processing is executed in *work* method. Typically, work method takes four arguments, number of input items, number of output items, the head pointer of the vector of input item and the head pointer of the vector of output items. The first two arguments are optional while the last two arguments are necessary.

- Number of input items is collected by GNU Radio framework from the previous signal processing block. As GNU Radio framework defined, every *work* method returns the number of output items. Thus, if the next block desires the number of input items, the returned number of output items from previous block will be available for calculation.
- Number of output items is calculated manually or by default. Generally, the signal processing algorithm requires the accurate number of output items. For example, the adder block determines the size of output buffer by calculating the number of output items.

- Head pointer of the vector of input item is the start point of a consecutively stored input stream. Based on the pointer, the block can retrieve all the input data and execute signal processing.
- Head pointer of the vector of output item the start point of a consecutive space for holding the output data. Typically, part or entire input data are copied to this space for the following calculation. For example, the adder block copies all the data from the first input port and execute addition on the data from all the input ports correspondingly.

GNU Radio framework defines independent thread for each block. When launch the system, every block is initialized and operates simultaneously. Take pipeline in a factory as example, the belt of each pipeline is working whenever product exists or not. Once the first pipeline obtains raw materials, the processed materials are passed to next pipelines until all the products are completed. Comparing to GNU Radio framework, the execution of the constructor of a class is the initialization of pipelines and the *work* method is the belt on each pipeline.

After initialization, the *work* method starts to work. The work flow of the *work* method is listed as follows.

1. The *Connect* method connects current and previous block and the number of output items and the output stream are available for current block.
2. Current block reads the number of output items of previous block as the number of input items and calculates the number of output items.
3. Current block reads the output stream of previous block as the input stream.

4. Pass the number of output items and input stream to the *work* method and start to process the signals.
5. Store the processed signals in the output buffer and return the number of output items.
6. The *connect* method collects the output stream and the number of output items for next block's use.

All of the blocks in the system keep operating until the system is shut down or the work load is beyond the performance of the hardware.

3.3.5 Tags and Messages

Tags and messages are two vital mechanisms associated with most of the signal processing blocks. Both of them transmit Meta data and information related to the data streams. However, tags are transmitted along with data stream, while messages are transmitted through a specific port. Figure 3.4 presents how tags works with data passed from one block to another. Assume that one letter represents a small amount of data. When data flows, the associated tags are transmitted simultaneously in parallel. Meanwhile, the information of each tag may vary from others. At the input port of next signal processing block, the tags are retrievable only if the position is known. In addition,

Figure 3. 4 Tags associated with some data at different positions on data stream

the accurate position is required for tag retrieving. For example, only knowing the position of data b, f and p on the data stream shown in Figure 3.5 can the signal processing block obtain the information stored in tags.

Unlike tags, messages occupy a special port of the signal processing block. Whenever the message is sent, the receiver block will call a bound function to deal with the information stored in the message. Generally, the information in a message stored in pairs with key and value. In the construct of OFDM and SC-FDMA modulation in GNU Radio, the tags and messages are widely used for Meta data transmission in order to simplify the flow graph and burden of programming.

Although the composition of signal processing block is not only what mentioned above, smart pointer, IO signature, different types of blocks, core of blocks and the mechanism of tags and messages are the essential components of signal processing block. Smart pointer simplifies the maintenance of blocks after creating objects for blocks; IO signature defines the input and output port, which is the foundation of communication; different types of block distinguish the models of signal processing; the work method conducts the execution of signal processing algorithm. As the trunk of a tree, these four components consist of the backbone of C++ programs of signal processing block in GNU Radio framework.

CHAPTER 4: PHYSICAL LAYER OF LTE IMPLEMENTED ON GNU RADIO

LTE standard adopts OFDM technology as the downlink modulation to provide large channel capacity and better anti multipath fading performance. However, the mobile devices require lower complexity and the control of power consumption. Hence, Single Carrier Frequency Multiplexing Access (SC-FDMA) technology is utilized in uplink communication. This chapter will explain the logic of the communication between base station and user devices. Then the implementation of OFDM and SC-FDMA modulation is introduced. The source codes are published to GitHub [21].

4.1 Communication Logic of User Device and Base Station

With the downlink and uplink communication systems, we are able to establish the communication between user devices and base station in GNU Radio. This section focuses on the establishment of communication by typical 3-way handshake model and the packet allocation and identification.

4.1.1 3-Way Handshake

Originally, 3-way handshake model is adopted in TCP/IP. For simplicity, this thesis implements the connection logic via 3-way handshake model. The procedure of 3-way handshake is shown in Figure 4.1. To establish a connection, the user device sends a request to base station with its device ID. Typically, the device ID is International Mobile Equipment Identity (IMEI), which is the unique sequence for each mobile device.

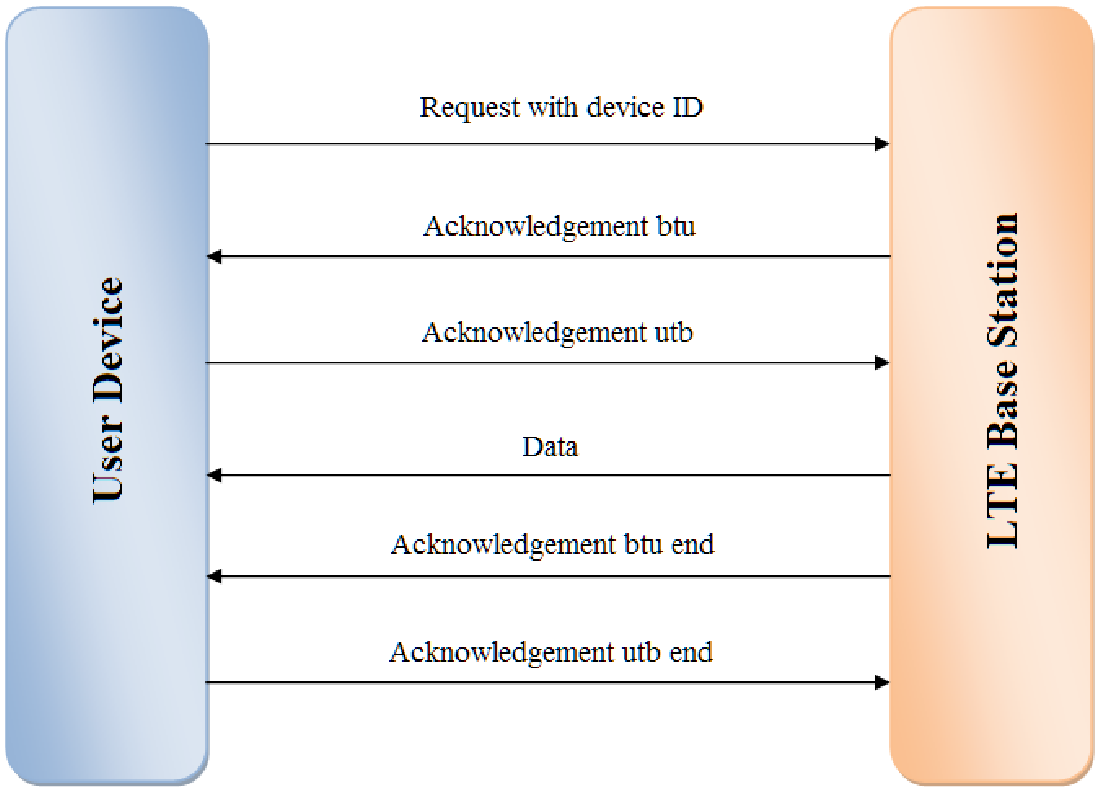


Figure 4. 1 Communication logic between user device and LTE base station

The base station will check the IMEI with Home Location Register (HLR) where the cellular network stores the information of each mobile device. If the device is roaming in other area, the Visitor Location Register (VLR) will allocate a Temporary Mobile Subscriber Identity (TMSI) which is an instead ID for roaming mobile devices. If the base station assures the device ID, it sends an acknowledgement to user device to confirm that it receives the request from the user device. After recognizing the acknowledgement from base station, the user device sends back another acknowledgement to base station for the confirmation of receiving the acknowledgement from base station. Finally, the base station starts to transmit data requested by the user device after identifying the answered acknowledgement from the user device. In conclusion, the request and acknowledgement are sent back and forth for 3 times before base station transmits data to the user device.

Additionally, we create a scheme for disconnection of the data transmission after the last data packet is transmitted. Showing in the Figure 4.1, the last two steps are the procedure of transmission disconnection between user device and base station. Once the last packet is transmitted, the base station sends an acknowledgement to the user device for the notice of the end of transmission. Then the user device sends an acknowledgement back to the base station if it receives the acknowledgement from the base station. Eventually, this session is ended after the base station receives the acknowledgement from the user device.

4.1.2 Packet Allocation and Identification

In this thesis, we assume the user device only send request and acknowledgement, which are small amount of data, to the base station, while the base station send data with high capacity, as well as acknowledgement with small size. Since the data modulation of uplink communication indicates that the signal transmits in serial, the whole bandwidth is occupied by one symbol at one time. Hence we do not consider about the subcarriers allocation for uplink communication. On the other hand, the identification of user devices on base station side utilizes the device ID in the header of each packet sent from user devices. The header structure is introduced in Table 4.2. The flow graph of the

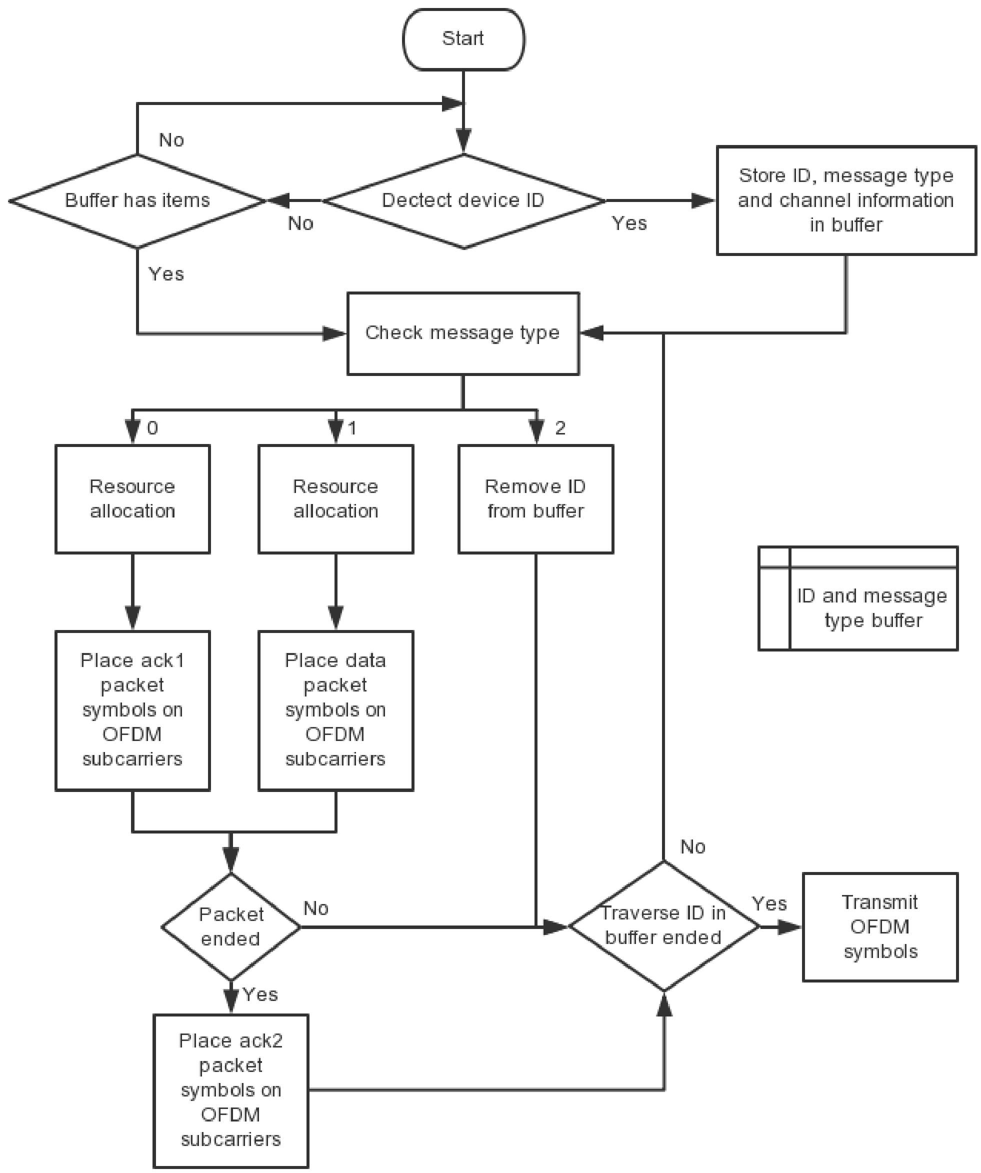


Figure 4. 2 The flow graph of the scheduler of the base station

implementation of packet allocation and Identification on base station side is shown in Figure 4.2. The heart of the base station is a scheduler which is in charge of user devices identification, dynamic resource allocation and packet dispatching. The dynamic resource

Table 4. 1: The representation of message types for packet from user device and base station

Message type	User device	Base station
0	Request	Acknowledgement btu
1	Acknowledgement utb	Data
2	Acknowledgement end utb	Acknowledgement end btu

allocation algorithms are introduced in Section 2.3.Previously, the base station creates a buffer for storing the ID and message type of the active user devices. If an ID exists in the buffer, the base station is transmitting acknowledgement or data to the corresponding user device. The message types of the packet sent from user devices is shown in Table 4.1. If the message type of the packet received by base station is 0, it indicates that the packet is a request from the user device; if the message type is 1, it indicates that the packet is an acknowledgement for confirmation that the user device receives the acknowledgement from the base station; the message type 3 implies that the packet is the acknowledgement for knowing the data transmission completed.

Now we explain the flow graph of the scheduler. First of all, the scheduler keeps detecting incoming signals. As long as a valid packet is received, the scheduler resolves the device ID and message type from the packet header. Then, it stores the ID and message type in the buffer and identifies the type of the packet. If the scheduler does not detect a valid packet, it will traverse the buffer to check if there is data for processing.

Based on the result of buffer checking, the scheduler may keep detect packet if no items in the buffer or it may keep processing data for each device ID.

Second, after figuring out the type of received packet, the scheduler traverses the buffer and processes the data for each device ID based on the message type. If the message type is 0, it places the symbols of the acknowledgement called "acknowledgement btu" on the subcarriers acquired from dynamic resource allocation algorithms; if the type is 1, the "acknowledgement utb" and the estimated channel information calculated by the user device based on the previously transmitted acknowledgement will be included in the incoming packet. Then it places data symbols on the allocated subcarriers; if the type is 2, it removes the ID and its associated information from the buffer since the data transmission is finished.

Third, the symbols are probably not transmitted to the user devices immediately after copied to the OFDM resource block. If there is more than one device ID in the buffer, the subcarriers are impossible to be entirely allocated to one user device because more than one user devices are receiving data from the same base station. Only if all of the available subcarriers are occupied by data symbols can the base station emits the signal to the air. Therefore, the schedule needs to traverse the buffer to guarantee the data for all of the user devices are placed on their own subcarriers. Once the OFDM resource block is ready for transmission, the scheduler adds an outer header for the OFDM resource block for distinguishing the subcarriers for reconstruction of data packet.

Fourth, if the last amount of data are place on OFDM resource block, the scheduler will transmit the acknowledgement named "acknowledgement end btu" to the corresponding user device. If the base station receives "acknowledgement end utb", the scheduler removes the device ID from the buffer, which means the transmission is completed.

In order to obtain the required data packet from the signals, the user device owns a decomposer for data screening according to the OFDM resource block structure. The flow graph of the decomposer is shown in Figure 4.3. Based on the outer header of the OFDM resource block, the decomposer is capable to distinguish the data on the subcarriers with the same carrier ID. The structure of the OFDM resource block is shown in Figure 4.4. According to the instance, the first 6-bit sequence on each subcarrier is the carrier ID and the outer header consists of all of the carrier IDs. The subcarrier 1, 3, 4, 6 have the same carrier ID, which implies that the data on these subcarriers are in the same data packet. If the performance of channel estimation is adequate for equalizer to perfectly recover the data, the decomposer is able to recognize and reconstruct the data packet from subcarrier 1, 3, 4, 6. Then, it separates the header of the data packet which contains the information of the packet such as device ID, packet length and so forth. Finally, the decomposer checks the resolved device ID to determine whether keep or abort the packet. If the device Id matches, the user device receives a data packet successfully, otherwise, it will not stop reconstructing data packet from the subcarriers with different carrier Id until it find the data packet belonging to this user device.

Once receiving a packet, the message type included in the packet header will guide the behavior of the user device. The relationship between message type and the

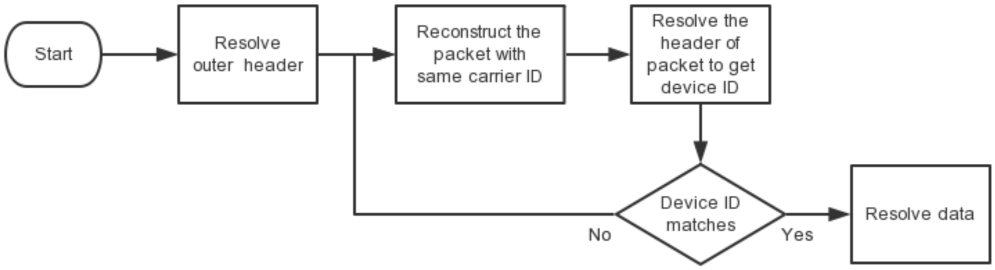


Figure 4. 3 The flow graph of the decomposer on user device

Carrier ID	Data
1 1 1 0 1 0	1 0 0 1 1 1 1 1 ......
1 0 1 0 0 0	0 0 0 0 0 1 1 1 ......
1 1 1 0 1 0	1 0 1 0 0 0 0 0 ......
1 1 1 0 1 0	1 1 1 1 1 1 1 1 ......
1 0 1 0 0 0	1 1 1 0 0 1 1 1 ......
1 1 1 0 1 0	0 0 0 0 1 1 1 1 ......

Figure 4. 4 The structure of OFDM resource block (symbols are converted to binary)

data in the packet is listed in Table 4.1. If the message type is 0, it indicates that the received packet is the acknowledgement which confirms that the request is received by the base station. Meanwhile, the user device appends the estimated channel information to the acknowledgement packet which will be sent to the base station for the completion of establishment of connection. If the message type is 1, the user device receives the requested data. If the message type is 2, the user device is informed that the last packet of requested data is transmitted.

In conclusion, this design for the communication logic between user device and base station is the minimal of the wireless communication in the real world. More aspects need to be taken into consideration to improve the performance of the wireless communication system. Based on the design, the implementation of the downlink and uplink is introduced in the following two sections.

4.2 Downlink Module

Orthogonal Frequency Division Multiplexing (OFDM) modulation is widely used in varieties of communication standards such as WiMAX and LTE standards. Nowadays the communication equipments which support these standards are ubiquitous since the mobile communication technologies are developing rapidly. This section introduces the basic theory of OFDM modulation first and then develops OFDM modulator and demodulator via GNU Radio for the communication between LTE base station and multiple users.

4.2.1 Introduction to OFDM

OFDM modulation is similar to Frequency Division Duplex (FDD) since they divide a wide range of frequency into some small range of frequencies and allocate these frequencies as channels to users. The difference is that OFDM allows these small ranges of frequencies overlap a specific amount but they do not interfere with each other, in other word, they are orthogonal with each other. In this case, the same bandwidth contains much more channels than FDD so that the channel capacity increases tremendously. The typical flow graph of OFDM modulation is shown in Figure 4.5.

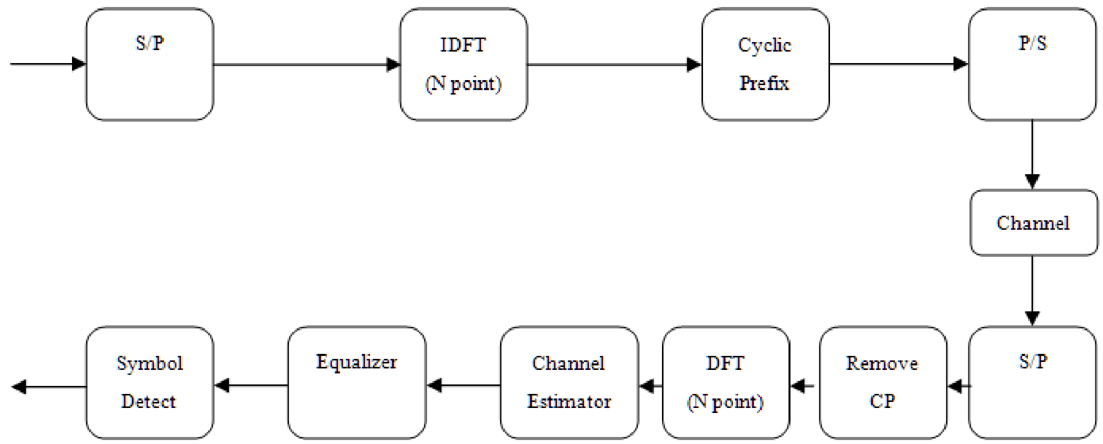


Figure 4. 5 Typical Flow Graph of OFDM Modulation

When the complex symbols come into OFDM modulator, the serial data are converted into parallel data which has N subcarriers transmitted simultaneously. Then compute IDFT on the parallel data and append cyclic prefix to each OFDM symbol to protect data symbols from Inter Symbol Interference (ISI). After converting the parallel data into serial, the data is transmitted to the receivers. The demodulator does the reverse of the actions of the modulator plus the work of recovery of the data from channel noise, multi-path fading and Doppler frequency shift. Generally, a channel gain estimator figures out the channel taps in frequency domain through known symbols or some blind estimation algorithms. Subsequently, the equalizer recovers the symbols using the estimated channel taps.

Although OFDM modulation contributes to more efficient bandwidth usage, the small bandwidth of each subcarrier with no guard frequency introduces no neglecting of

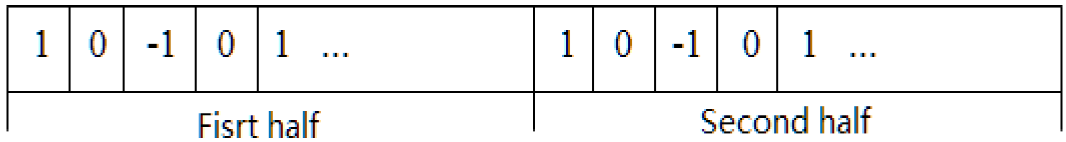


Figure 4. 6 The Preamble Sequence Used for Timing Synchronization by Schmidl and Cox (SC) algorithm

Inter Channel Interference (ICI) when frequency shift occurs. The sensitivity of frequency shift of OFDM modulation has motivated a large amount of researches on frequency offset estimation and the algorithms for synchronization. In this thesis, we utilize a mature algorithm developed by Schmidl and Cox. Another disadvantage of OFDM modulation is high peak-to-average power ratio (PAPR). Since the subcarriers are transmitted in parallel, the transmitted signal is the sum of all the signals on each subcarrier, which leads to some high peaks on the signal but the average amplitude keeps relatively low. Considering about the cost and power control on mobile devices, LTE standard applies OFDM modulation only for downlink communication.

4.2.2 Timing Synchronization and Frequency Correction

Unlike MATLAB simulation for OFDM based system, the communication system in the real world require more consideration for signal transmission. The signal synchronization and frequency correction is one of the most important issues on the receiver devices. For signal synchronization and frequency correction at the frontend of receiver, GNU Radio provides a block using Schmidl and Cox (SC) algorithm [4]. SC algorithm uses preamble symbols created by Pseudo Noise (PN) sequences to detect the start point of OFDM symbol on the receiver devices. The preamble sequence is designed as Figure 4.6 shows. The sequence consists of two halves and the PN sequences are placed only on odd positions. Besides, the sequences on the first half and second half are the same. The core idea of SC algorithm is to compute the correlation of the halves of the preamble symbols. Let R(d) be the energy of received signal, P(d) be the conjugate multiplication result and M(d) be the timing metric that is used to detect the start point of received symbols. The energy is

$$R(d) = \sum_{k=0}^{N/2-1} | r(d+k+\frac{N}{2}) |^2 \qquad (4.1)$$

and the conjugate multiplication of the preamble sequence is

$$\mathrm{P}(d) = \sum_{k=0}^{N/2-1} r^*(d+k)\,\mathrm{r}(d+k+\frac{N}{2}) \qquad (4.2)$$

where d is the start position of the sliding window, k is the amount of shift of the window and N is the length of the preamble sequence.

R(d) is the energy of the half of the preamble sequence. P(d) calculates the conjugate multiplication of shifting half preamble sequence and the other half. Finally, the timing metric is measured by

$$M(d) = \frac{|P(d)|^2}{(R(d))^2} \qquad (4.3)$$

which is the correlation of halves of preamble symbols. Ideally, M(d) should be 1 if the shifting part enters the range of symbols without ISI, which shows as a plateau in the timing metric. However, if the noise exists, the height of the plateau will be lower than 1 (see Figure 4.7). When the correlation drops, the dropping position is the start point of the

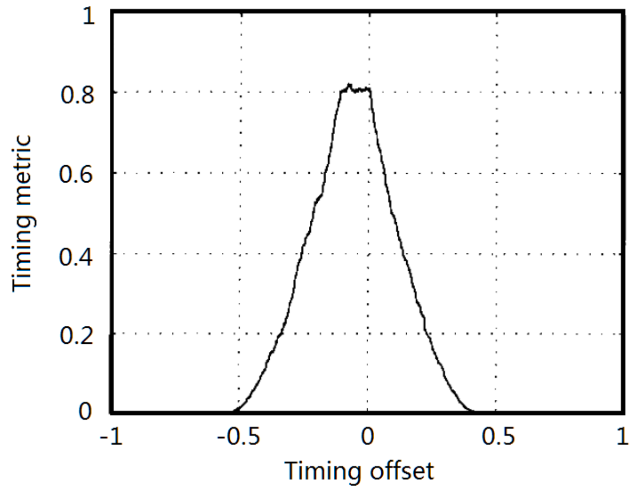


Figure 4. 7 The timing metric M(d) with AWGN channel (SNR = 10dB) [4]

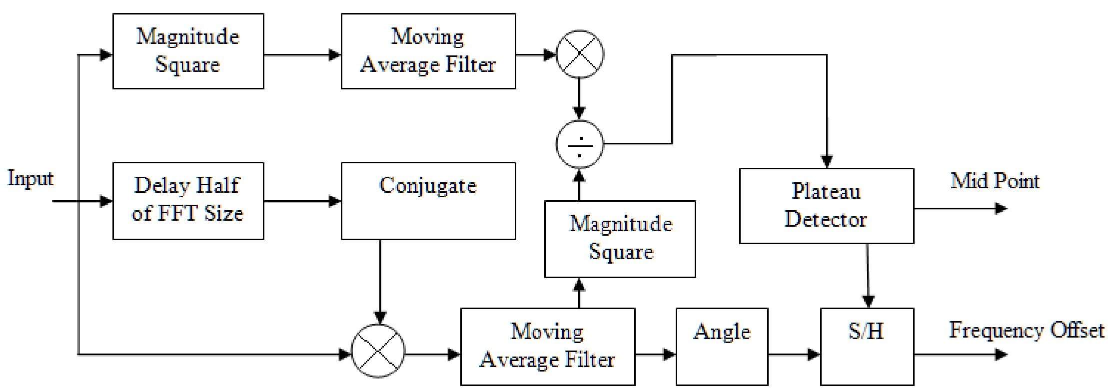


Figure 4. 8 The structure the implementation of Schmidl and Cox Timing synchronization and frequency offset estimation algorithm in GNU Radio

data signal. The other part of SC algorithm is frequency offset estimation. The frequency offset is done by calculating the angle between the conjugated first half of the preamble sequence and the second half. The implementation structure of SC timing synchronization algorithm is shown in Figure 4.8. The original solution contains a matched filter after the division operator to help detecting the drop edge of the plateau.

However, the plateau is located in the cyclic prefix symbols without ISI. Thus, all of the effective received symbols are preserved and can be recovered by the estimated channel taps.

4.2.3 Implementation of OFDM by GNU Radio

GNU Radio provides a simple example of OFDM modulator and demodulator flow graph. However, this example only supports the communication between one user device and one base station. In order to support multi-user communication, some required block creation and modification are completed for establishment of the communication between one base station and more than one user devices. Figure 4.9 shows the structure of OFDM modulator that the base station sends data to user devices. Figure 4.10 shows the structure of OFDM demodulator that one user device receives the data from the base station. All of the blocks in OFDM modulator are explained as follows.

- **Base station core**

  This block detects the user devices. It sends acknowledgement or requested data to the user devices if it receives request or acknowledgement from user devices. To process the request and send data stream to multiple user devices, it contains a scheduler to manage the data which belong to different user devices. The scheduler is explained in detail in Section 4.3. The dynamic resource allocation algorithms are implemented in this block.

- **Stream CRC32**

This is a built-in block of GNU Radio. It generates 4 bytes Cyclic Redundancy Check (CRC) according to the data packet. The 4 bytes are attached to the end of data packet.

- **Packet header generator**

  The original packet header generator only contains the packet length, packet ID and an 8-bit tiny CRC, which is only compatible with one-to-one communication. In order to support multi-user communication, the header structure is designed in Table 4.2. The use of message type is mentioned in Section 4.3.

- **Repack bits**

  This is a built-in block of GNU Radio. It converts some bits into an integer for the preparation of constellation mapping. For instance, this thesis uses QPSK to map the payload stream. Thus, this block converts every 2 bits into an integer from 0 to 3.

Table 4.2 Packet header structure for multi-user communication

Items in header	Number of bits
Device ID	16
Message type	2
Packet length	11
Packet number	11
CRC	8

- **Chunks to symbols**

  This is a build-in block of GNU Radio. It maps the integers from repack bits block to varieties of constellation symbols or directly from other blocks that the data will be mapped to BPSK symbols. For example, the integers from 0 to 3 are mapped to QPSK symbols.

- **Tagged stream mux**

  This built-in block merges the header symbols and payload symbols.

- **OFDM carrier allocator**

  This block is modified from the original one provides by GNU Radio for working with the scheduler of the base station core. It gather the packets with different device ID and allocates them on specific subcarriers according to the results of dynamic resource allocation algorithms. Before sending symbols to next block, it adds two synchronization words for time synchronization and frequency offset estimation. Moreover, the second synchronization word also acts as the training sequence for initial channel estimation.

- **OFDM cyclic prefixer**

  This build-in block copies the last N symbols from each OFDM symbol and paste to the front of it. Typically, N is equal to the quarter of the amount of subcarriers.

Main blocks in OFDM demodulator are explained as follows.

- **Schmidl & Cox OFDM sync**

This build-in block implements SC algorithm to obtain the timing metric and frequency offset. Although the algorithm cannot detect the exact start symbol of required data, the frame equalizer will recover the correct order of symbols based on the estimated channel taps. After calculation the timing metric, it sends the trigger to inform the next block where is the start symbol of a packet.

- **Header/payload demux**

  This build-in block plays an critical role in received data processing. First, it checks the trigger signal from the Schmidl & Cox OFDM synchronization block. Once it finds the trigger, the synchronization words and outer header will be transmitted first. After receiving the information from the parsed outer header, it transmits the rest of data along with the information to the second frame equalizer for the identification of the data belonging to this user device.

- **Channel estimator**

  Based on the second synchronization word, this block calculates the initial channel taps and sends them to the rest of blocks by tags. Before transmitting the outer header symbol, it removes both of the synchronized words.

- **OFDM frame equalizer**

  There are two frame equalizers, one works on the outer header symbols and the other one works on data packets. The one follows the channel estimator does equalization on the outer header symbols and transmits them to the serializer. The other one takes heavy burden work on separating the packet according to the subcarriers with the same packet ID. After reconstructing each packet, it parses the header of each packet and obtain the device ID. Comparing to the device ID of the current user device, this block only passes the data belonging to the valid device.

- **OFDM serializer**

  This block converts parallel transmitted data to serial based on the information of occupied carriers.

- **OFDM outer header parser**

  This block analyzes the outer header and figure out the index of subcarriers with the same packet ID. Then it informs the second frame equalizer for packet reconstruction.

- **User device core**

  This block simulates the behavior of terminal devices such as mobile phones. Initially, it sends request to base station with its device ID. Then it answers the acknowledgement and prepares for receiving required data from the base station. The logic of this block is described in detail in Section 4.3.

4.3 Uplink Module

Single Carrier Frequency Division Multiple Access (SC-FDMA) is accepted by 3GPP for the use of uplink transmission in LTE standard. This section introduces the theory of SC-FDMA modulation and compares SC-FDMA to OFDM modulation. Finally, the implementation in GNU Radio is introduced.

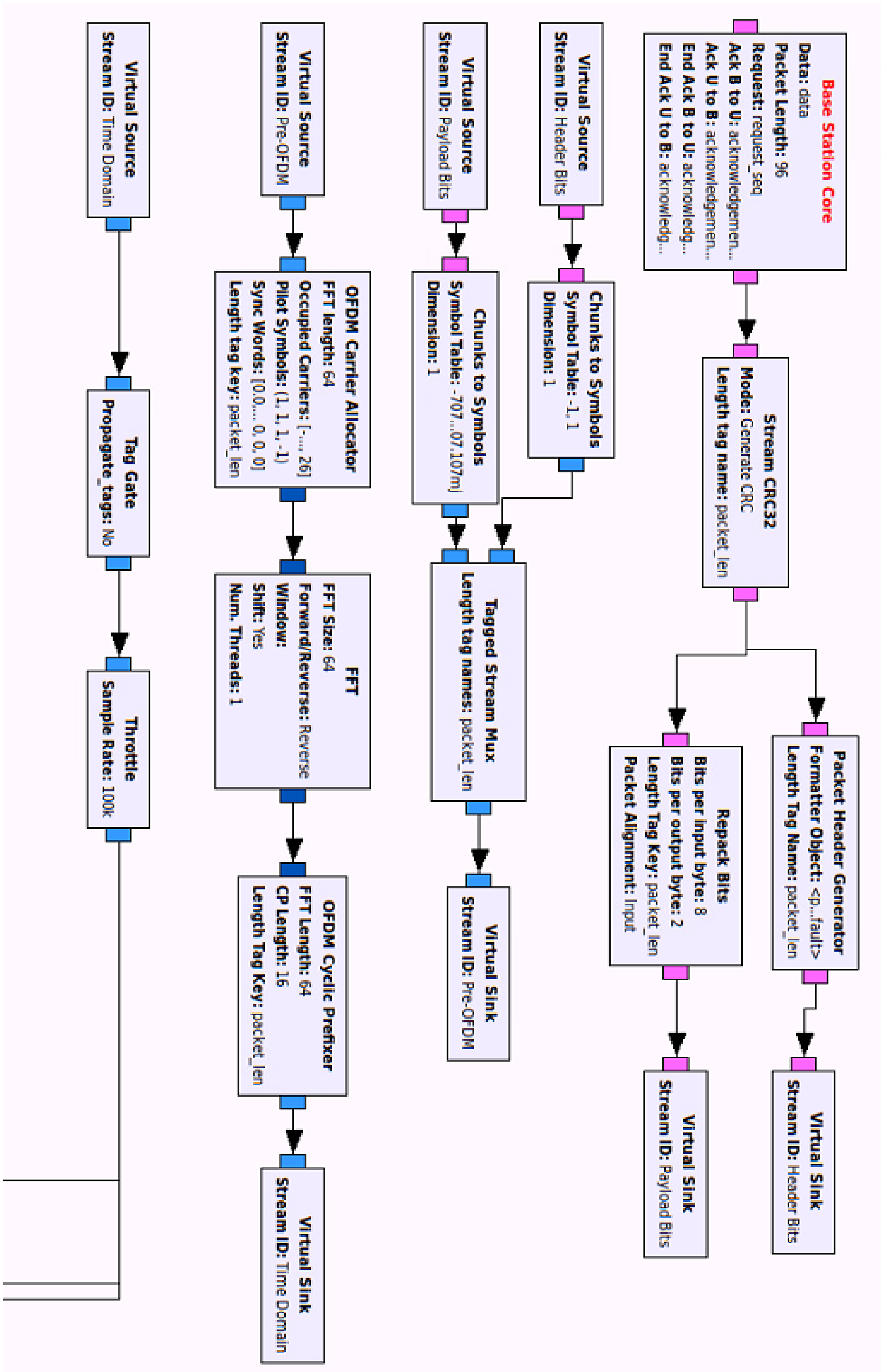


Figure 4. 9 The flow graph of OFDM modulator built by GNU Radio

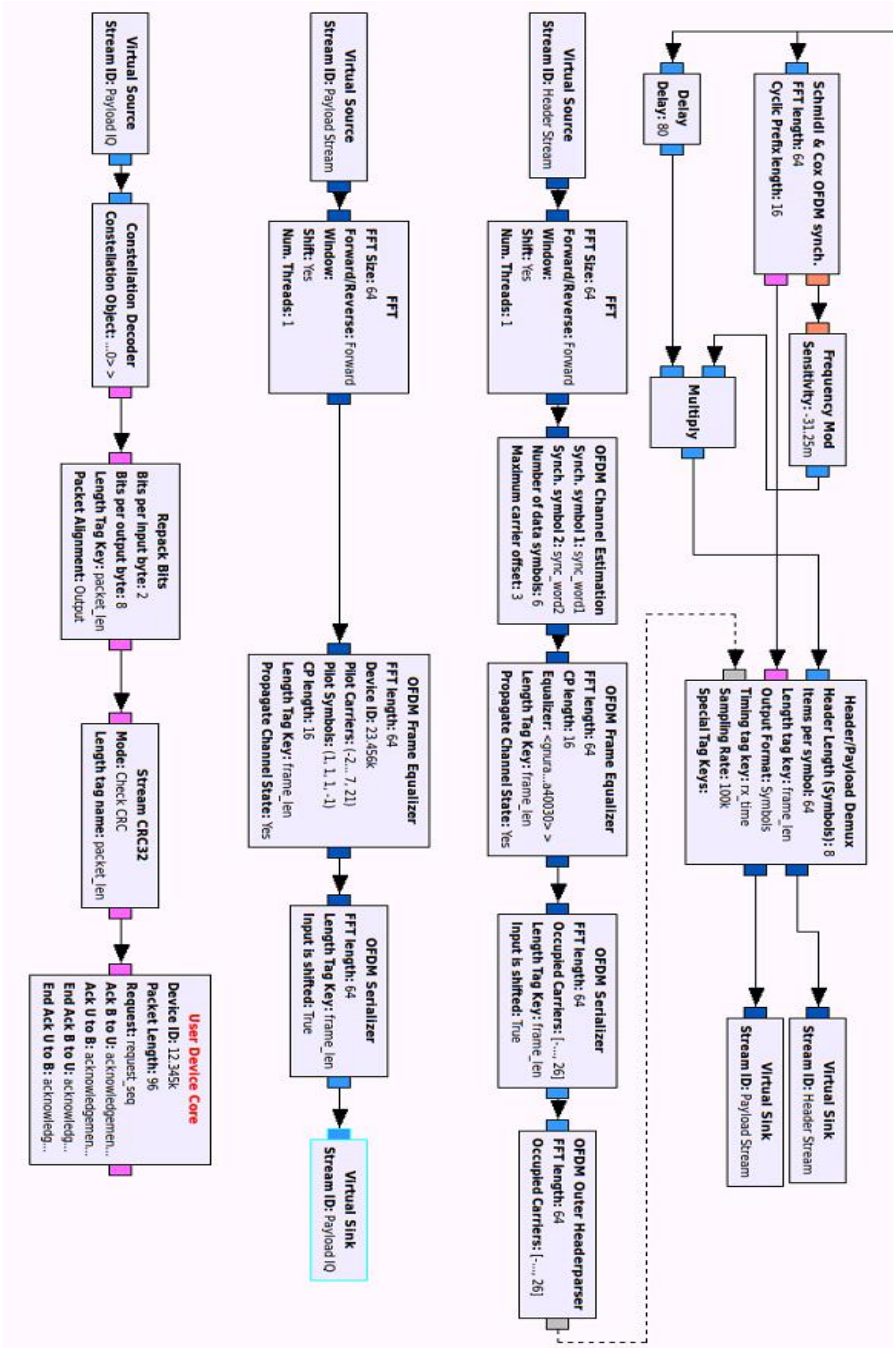


Figure 4. 10 The flow graph of OFDM demodulator built by GNU Radio

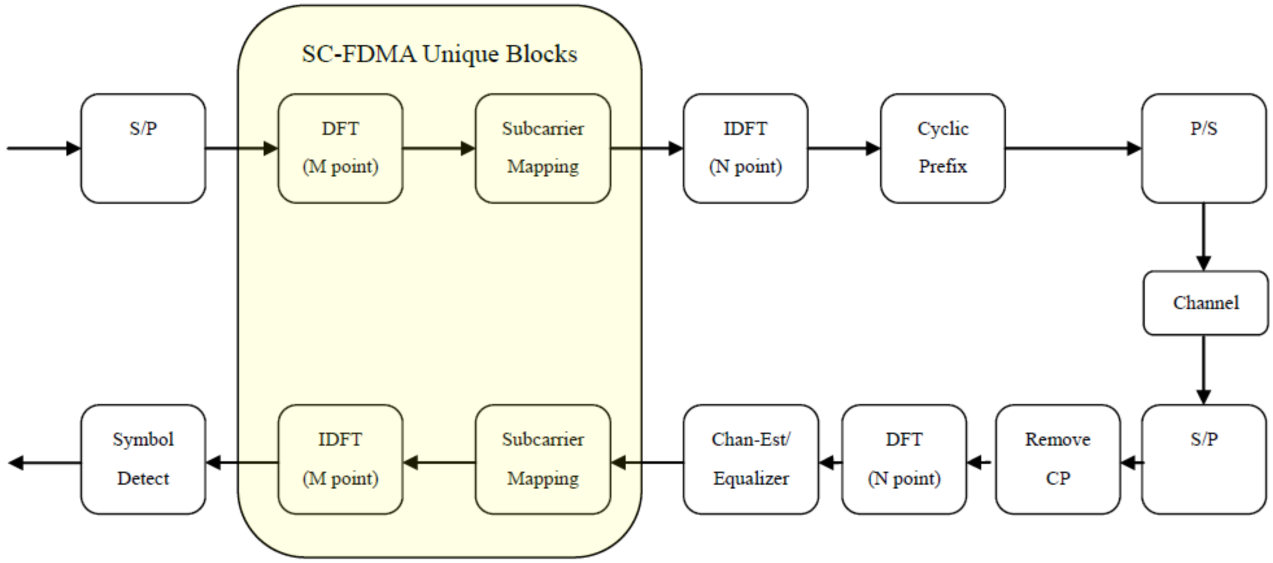


Figure 4. 11 Flow Graph of SC-FDMA Modulation

4.3.1 Introduction to SC-FDMA

As we know, OFDM technology transmits data in parallel, while SC-FDMA transmits data sequentially over the whole bandwidth which is why it is called single carrier FDMA. Although OFDM modulation improves the utility of spectrum and raises the capacity of channel, the high PAPR is still not ignorable. Since each subcarrier of OFDM modulation come into SC-FDMA modulator, the S/P converter changes the data to parallel with M time domain symbols. Then the parallel symbols are converted to frequency domain by M-point DFT. After placing the M parallel symbols to N subcarriers (M<N), the N-point IDFT converts data to time domain again. The rest part of the modulator is the same as OFDM modulator. Similar to OFDM demodulator, SC-FDMA demodulator adds a reverse subcarrier mapping and M-point IDFT after the equalizer. Since SC-FDMA modulation is the same as OFDM modulation but pre-

processes the data by a smaller point DFT and maps it to N subcarriers, SC-FDMA modulation is also named DFT-precoded OFDM or DFT-spread OFDM modulation.

There are two ways of subcarrier mapping: localized and distributed.

- **Localized mapping**

  Localized mapping places frequency domain data together over a wide bandwidth (Figure 4.12a). Suppose we have 4 symbols in a data block, they are carried by 4 consequent subcarriers on a wide bandwidth with 16 subcarriers and the rest of subcarriers are filled with 0. After IDFT, symbols are distributed to the whole bandwidth. Every two symbols are similar to be interpolated 3 symbols between them.

- **Distributed mapping**

  Distributed mapping disperses frequency domain symbols to the entire bandwidth (Figure 4.12b). Take the same 4-symbols data block as example, they are carried by 4 subcarriers with 3 subcarriers gap between each other. All the gap subcarriers are filled with 0. After IDFT, original data symbols are repeated in time domain over the symbol (OFDM symbol) period. Basically, distributed subcarrier mapping is simply implemented by interleaver.

In conclusion, the structure of SC-FDMA and OFDM is similar but the data contained on a symbol is different. Both of SC-FDMA and OFDM divides a wide bandwidth into small pieces and each piece carries data. However, the subcarriers of OFDM are allocated to different data such as the data with different sources or varieties of modulation, while all subcarriers of SC-FDMA contain one data symbol (Figure 4.13).

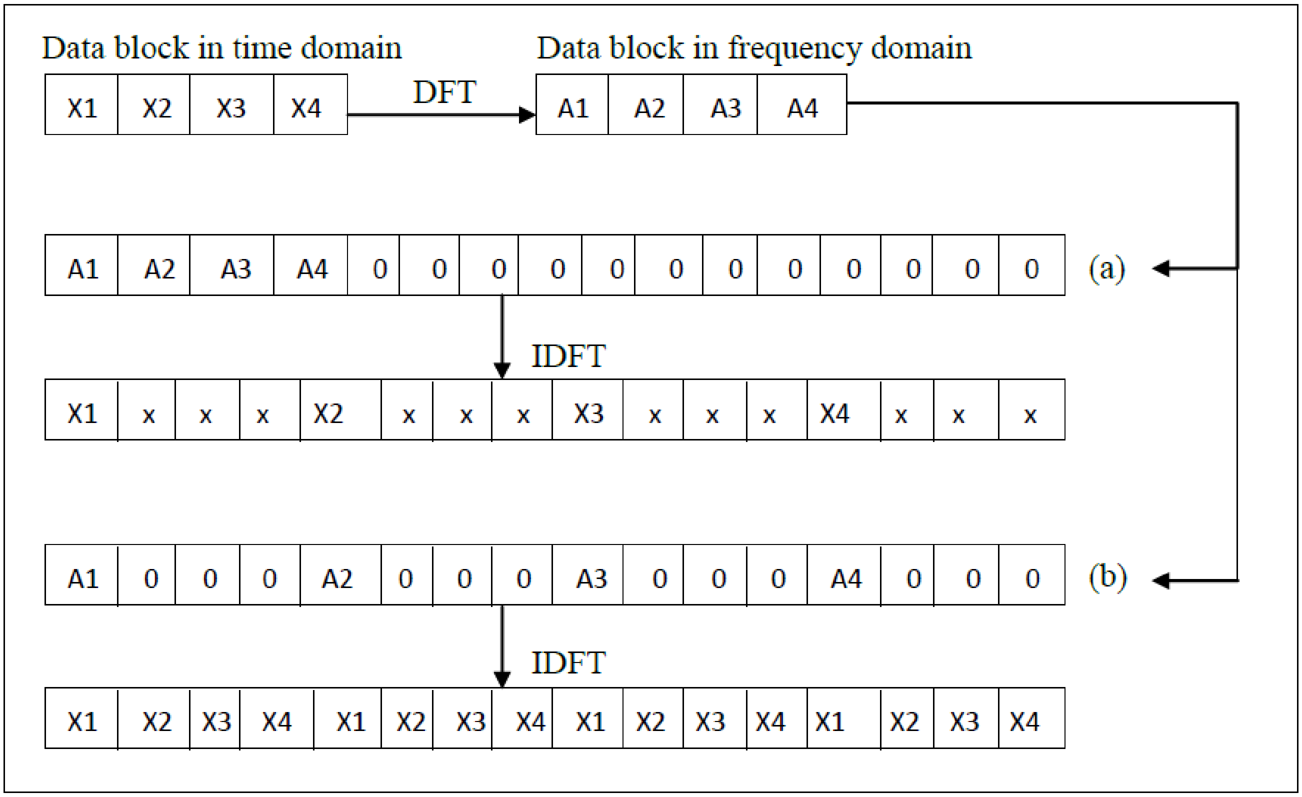


Figure 4. 12 Subcarrier Mapping Methods: (a) Localized; (b) Distributed

Therefore, each SC-FDMA symbol is the original data with same modulation like QPSK or 64QAM. As a result, the sum of the power of signals on all subcarriers remains even and low level, which means a low PAPR level.

4.3.2 Implementation of SC-FDMA by GNU Radio

The implementation of SC-FDMA is similar to that of OFDM. Most blocks from OFDM modulator are reserved but the essential modification is done for the SC-FDMA modulator. Since SC-FDMA modulation is the same as OFDM except for an additional

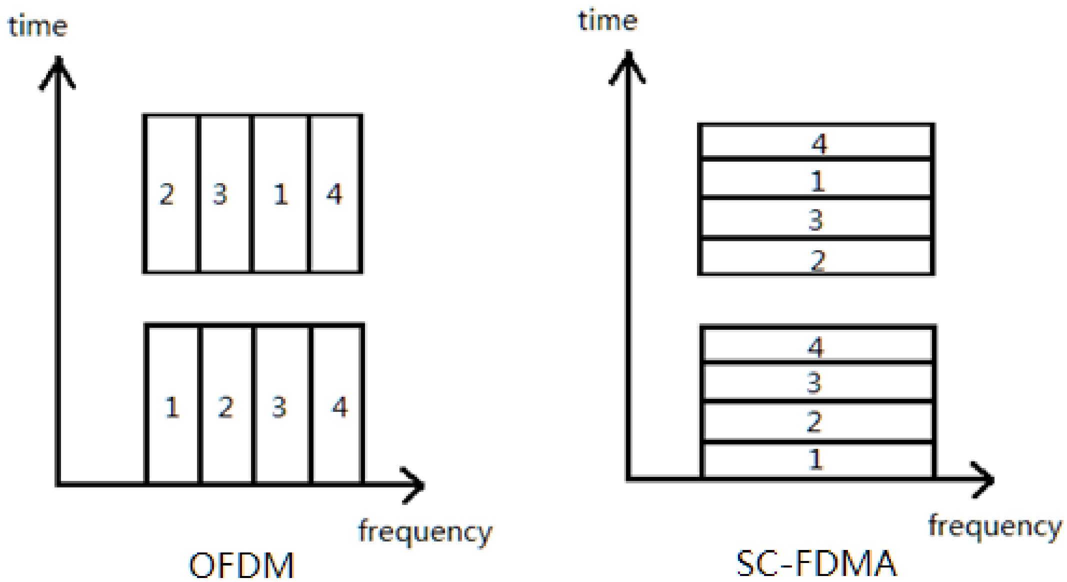


Figure 4. 13 Different symbol structures of OFDM and SC-FDMA

DFT and subcarrier mapping, two new blocks are created: SC-FDMA Carrier Allocator and SC-FDMA Interleaver. These two blocks are added before the reverse FFT block. Between the two new blocks, a forward FFT block is inserted. Thus, the SC-FDMA modulator is complete (Figure 4.13). The additional blocks are introduced as follows.

- **SC-FDMA Carrier Allocator**
- This block modifies the original OFDM Carrier Allocator. As the work is mapping original consequent data to parallel, the properties like pilot symbols, pilot carriers and synchronization words are removed. FFT length stands for the value of M mentioned in Section 4.2.1. The occupied carriers property covers all of the M subcarriers.

- **FFT**

  Since we need to execute forward FFT after S/P converter, this FFT block is set as Forward. Besides, the FFT length is the same as the value of M. Other properties remain the same.

- **SC-FDMA Interleaver**

  The interleaver, acting as subcarrier mapping, distributes the frequency domain data symbols to the larger set of subcarriers. The distribution scheme is based on Figure 4.11b. The lower FFT length stands for M and the upper FFT length stands for N mentioned in Section 4.2.1. The SC-FDMA demodulator is also similar to OFDM demodulator but the equalizer is simpler and the symbol decision is separated from the equalizer. The packet header structure is the same as it in OFDM modulation; however the packet header parser is independent to other blocks. The newly created and modified blocks are explained as follows.

- **SC-FDMA frame equalizer**

  Since there is no consideration about the packet distribution from user device to base station, the complicated logic and algorithm for packet identification is unnecessary in this block. Besides, the symbol decision is executed after the deinterleaved data with FFT computing. Hence, the equalizer and symbol decision are separated.

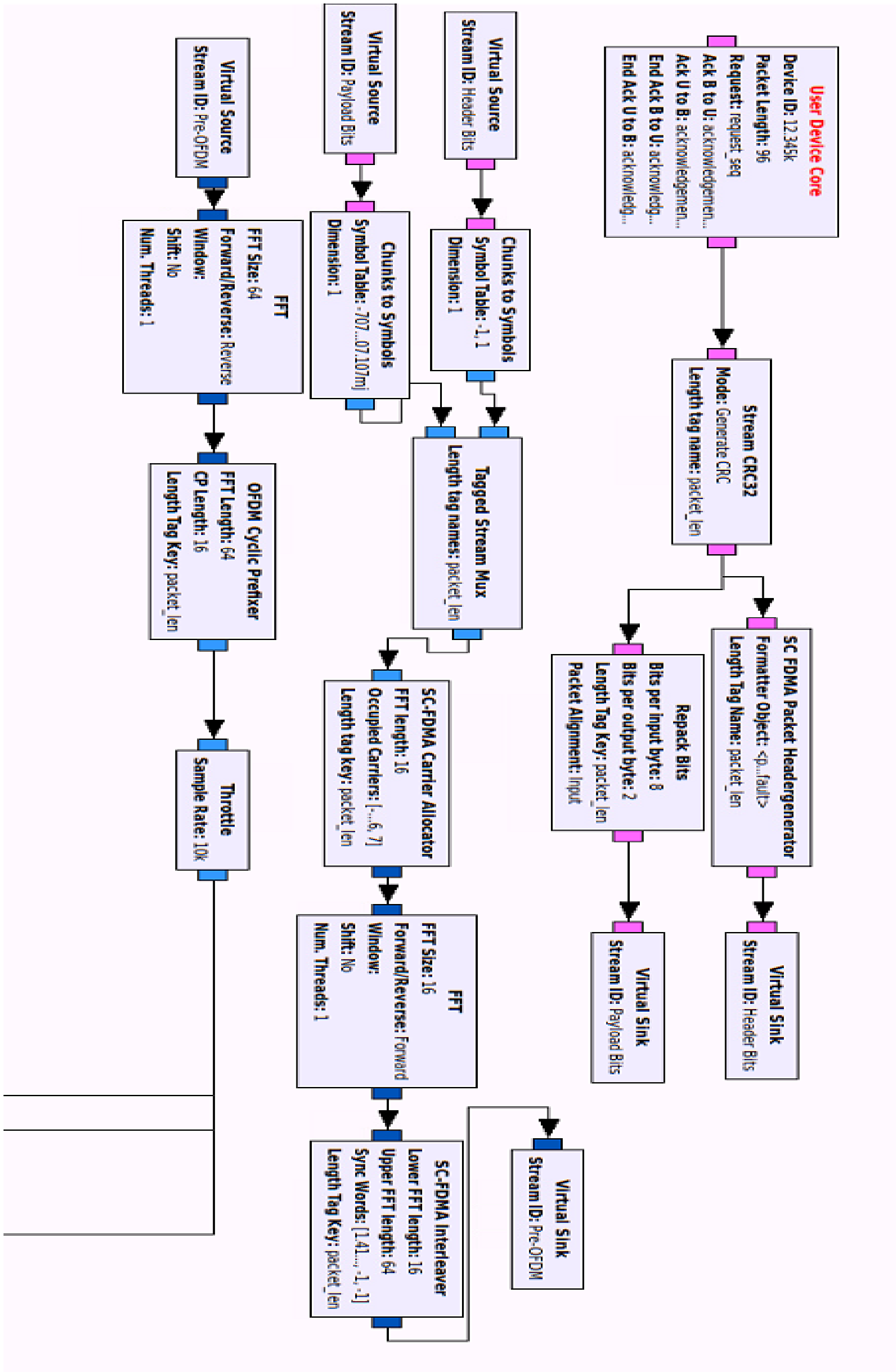


Figure 4. 14 The flow graph of SC-FDMA modulator built by GNU Radio

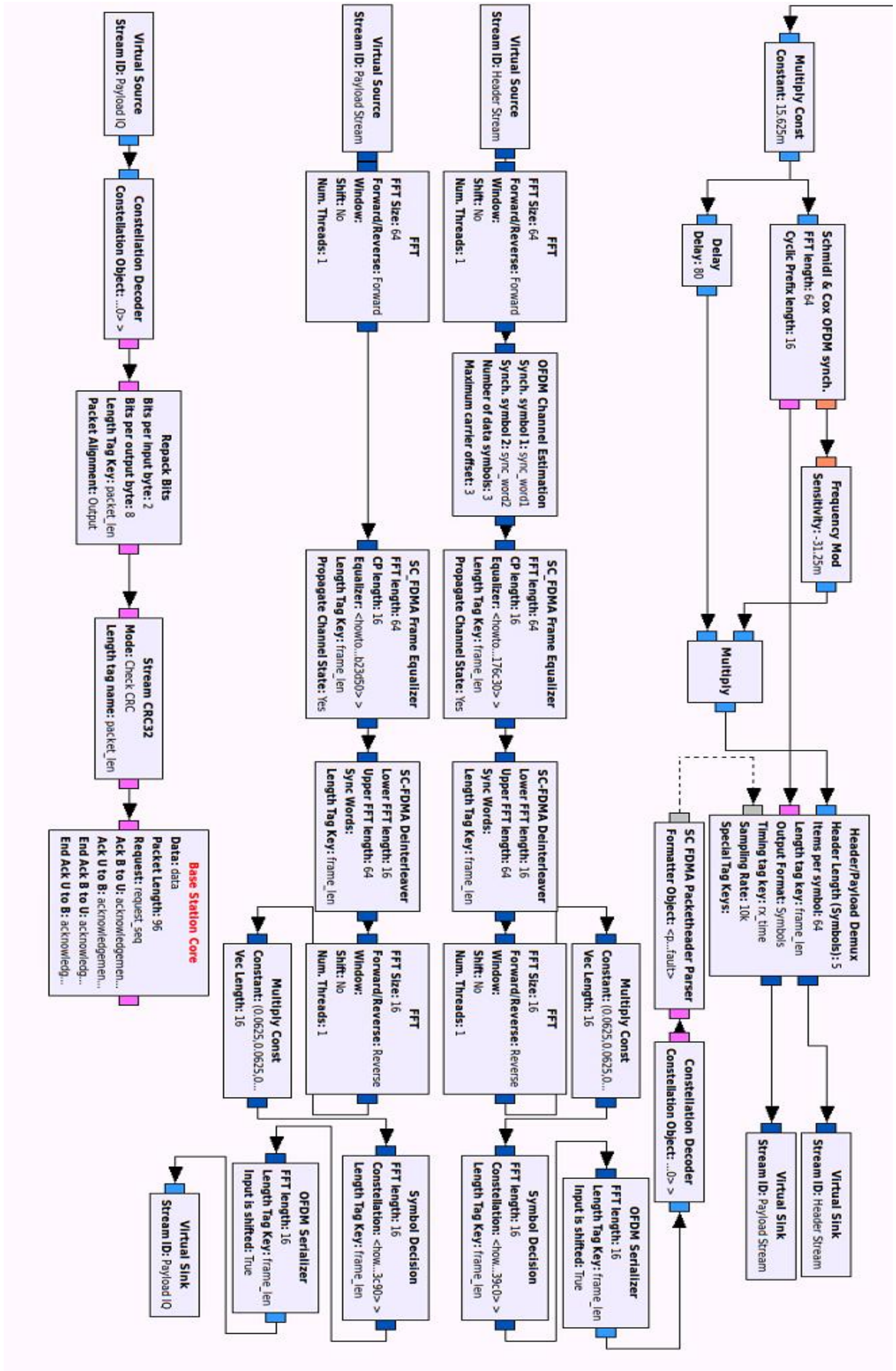
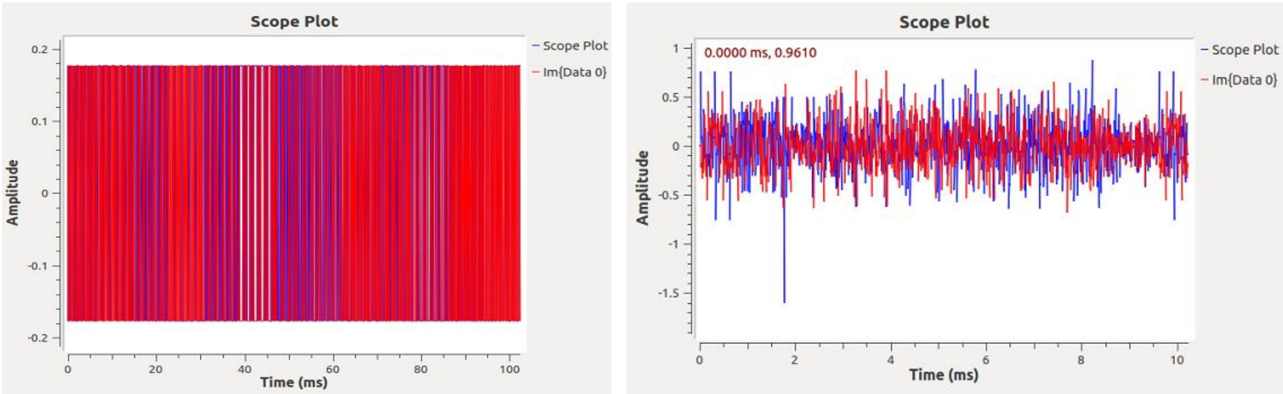


Figure 4. 15 The flow graph of SC-FDMA demodulator built by GNU Radio



(a) SC-FDMA (b) OFDM

Figure 4. 16 The Waveform of Transmitted Signal of SC-FDMA and OFDM

- **Symbol decision**

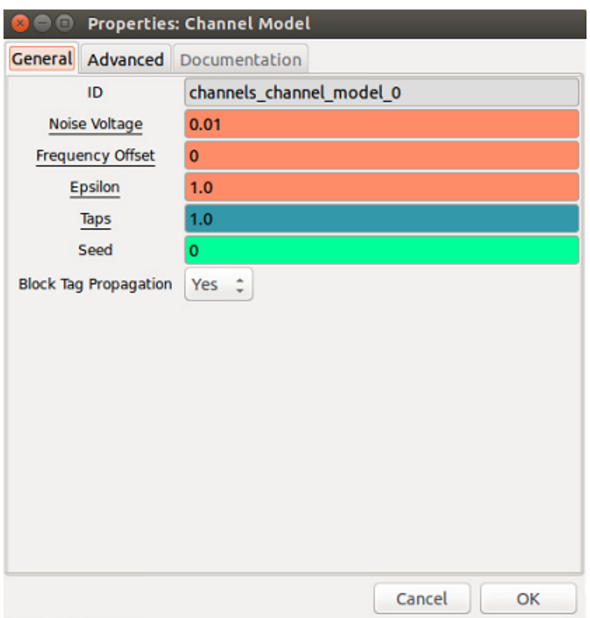
  This block converts the complex symbols to binary data according to the constellation mapping. In this thesis, we use BPSK on packet header and use QPSK on payload data.

4.2.3 PAPR Comparison between OFDM and SC-FDMA

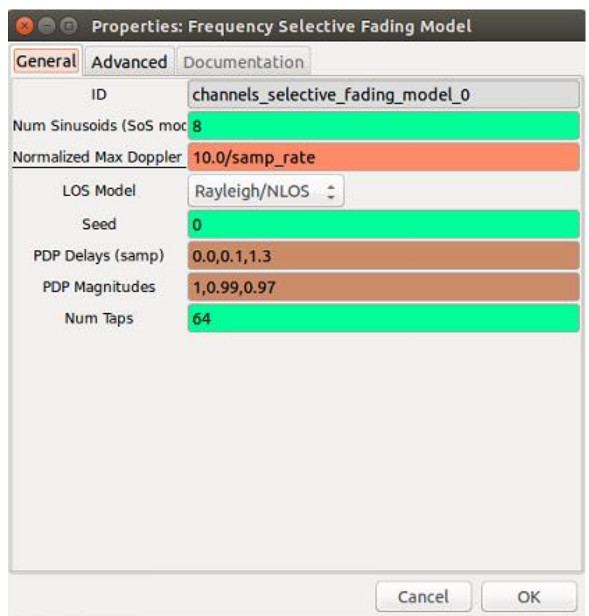
Typically, the transmitted signal of OFDM is shown in Figure 4.16b. Within the known timeline, we can see most part of the waveform keeps under the amplitude 0.5. However, there exist some peaks with amplitude over 0.5, especially the peak with much higher amplitude at 1.6ms. Thus, the PAPR remains a high level after OFDM modulation. Unlike OFDM transmitted signal waveform, the peak of waveform of SC-FDMA transmitted signal stays almost in the same level, which contributes to a close value of peak and average. Therefore, the PAPR of SC-FDMA modulation is quite lower than OFDM modulation.

CHAPTER 5: SIMULATION AND RESULT DISCUSSION

Since the limit of time and heavy burden of programming, this thesis simulates the system without USRP devices. In other words, this simulation uses the channel models provided by GNU Radio. In this case, the simulation is similar to MATLAB but using C++ and Python instead of MATLAB language. However, this work is the vital preparation for the simulation based on real world channel. The work on real world channel will be done later.



(a) AWGN channel



(b) Multi-path fading channel

Figure 5. 1 The channel model for the simulation

Table 5. 1 Simulation platform

Items	Version
Operating System	Ubuntu 12.04 LTS
GNU Radio	3.7.5.1
g++	4.6.3
Python	2.7

5.1 Simulation Model and Environment

The simulation platform is listed in Table 5.1. The whole system consists of a signal processing module named "gr-howto", two GRC flow graph files named "full-ofdm.grc" and "uplink.grc", and some C++ blocks provided by GNU Radio, which are explained in Chapter 4. The source code is updated to GitHub and share to public [21]. After compiling all of the C++ blocks, the system can be launched by the Python programs generated by the two GRC flow graph files. The simulation in "full-ofdm.grc" focuses on the LTE downlink communication, while that in "uplink.grc" focuses on the LTE uplink communication.

This simulation focuses on the dynamic resource allocation and the performance of the LTE downlink connection. The channel information is shown in Figure 5.1. Figure 5.1a is the AWGN channel model with noise voltage 0.01, while Figure 5.1b is the multi-path fading model with Rayleigh distribution and a small normalized maximum Doppler shift. The bandwidth is set as 1 MHz.

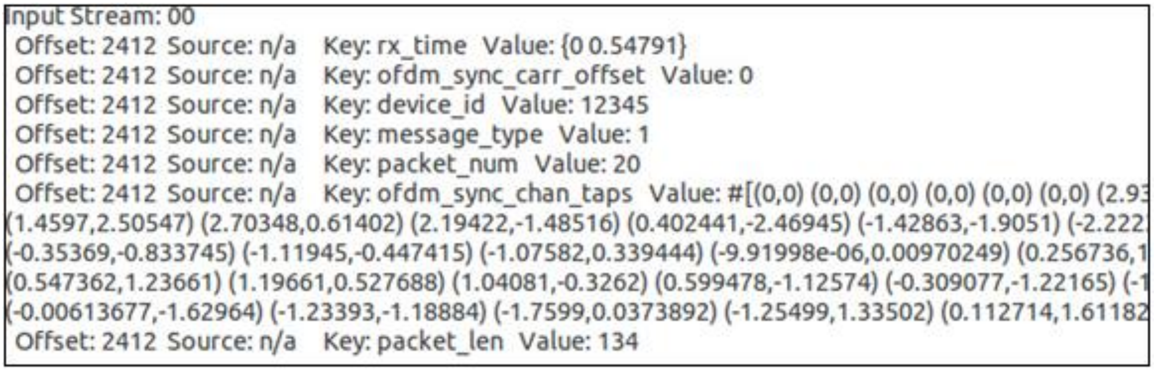
For the uplink, we only check the functionality since we do not apply dynamic resource allocation algorithms for uplink. Although the water-filling algorithm is best for single user, one SC-FDMA symbol contains just one data symbol over the entire bandwidth. Thus, implementing water-filling algorithm gains the complexity of the uplink connection. For the downlink, except for functionality checking, we apply all of the three dynamic resource allocation algorithms on the base station block. For the functionality check, we assign 64 subcarriers for the transmitted signal. In order to present the result in a clear way, we assume the number of subcarrier N = 8 and the number of users M = 3. The results of subcarrier and power allocation are collected from GNU Radio and plot in MATLAB.

5.2 Results and Discussion

The results and discussion is divided into two parts: uplink and downlink functionality check and the dynamic resource allocation analysis.

5.2.1 Uplink and Downlink Functionality Check

Both of uplink and downlink system introduce the 32-bit Cyclic Redundancy Check (CRC32) to determine whether the received packet is original one or not. Since the conflict probability of CRC32 is ignorable, we assume that the received packet is original if it passes the CRC32 check. When we send packets from user device to base station, the base station core block resolves a list of information shown in Figure 5.2. The key "rx_time" is the time point that the base station core receives the packet; "ofdm_sync_carr_offset" indicates the amount of the frequency shift; "ofdm_sync_chan_taps" shows the channel gains; "device_id" presents the source of the



```
Input Stream: 00
 Offset: 2412 Source: n/a   Key: rx_time  Value: {0 0.54791}
 Offset: 2412 Source: n/a   Key: ofdm_sync_carr_offset  Value: 0
 Offset: 2412 Source: n/a   Key: device_id  Value: 12345
 Offset: 2412 Source: n/a   Key: message_type  Value: 1
 Offset: 2412 Source: n/a   Key: packet_num  Value: 20
 Offset: 2412 Source: n/a   Key: ofdm_sync_chan_taps  Value: #[(0,0) (0,0) (0,0) (0,0) (0,0) (0,0) (2.93
(1.4597,2.50547) (2.70348,0.61402) (2.19422,-1.48516) (0.402441,-2.46945) (-1.42863,-1.9051) (-2.222
(-0.35369,-0.833745) (-1.11945,-0.447415) (-1.07582,0.339444) (-9.91998e-06,0.00970249) (0.256736,1
(0.547362,1.23661) (1.19661,0.527688) (1.04081,-0.3262) (0.599478,-1.12574) (-0.309077,-1.22165) (-1
(-0.00613677,-1.62964) (-1.23393,-1.18884) (-1.7599,0.0373892) (-1.25499,1.33502) (0.112714,1.61182
 Offset: 2412 Source: n/a   Key: packet_len  Value: 134
```

Figure 5. 2 The list of information printed in the console

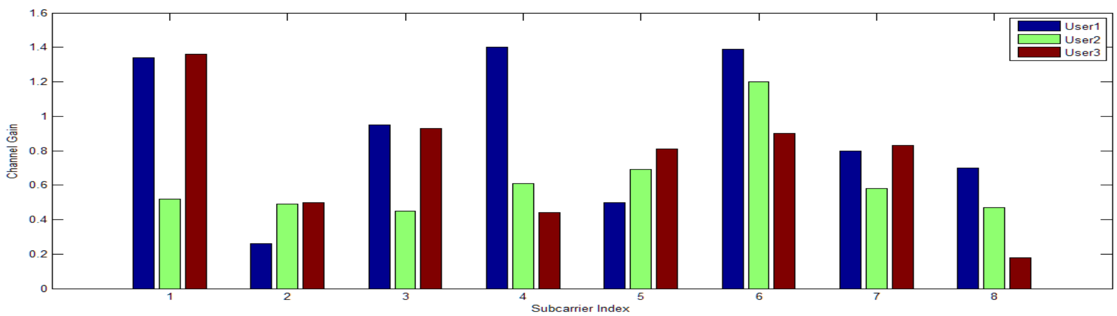
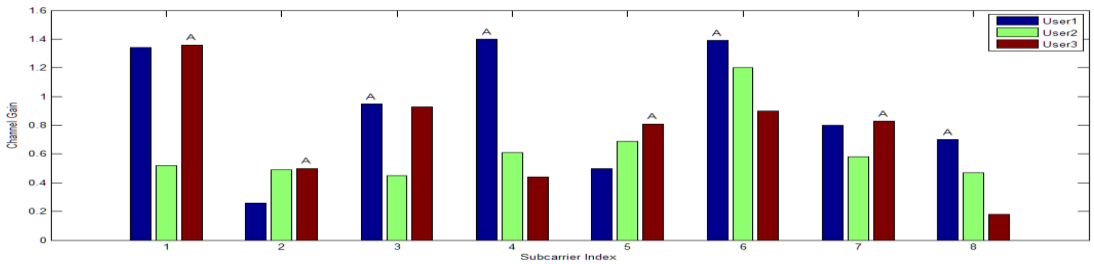


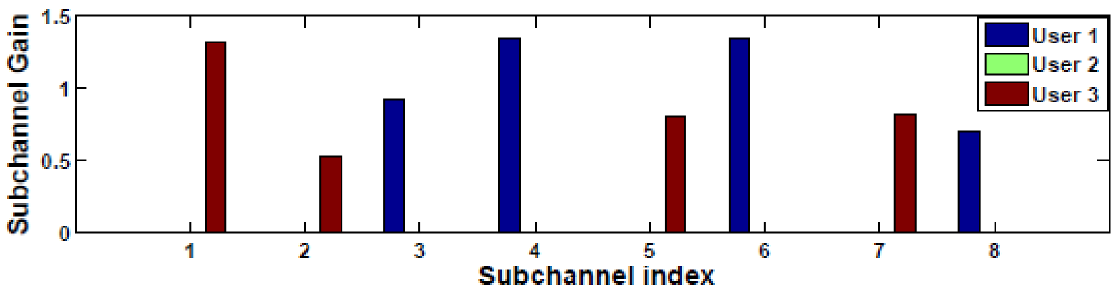
Figure 5. 3 Channel gains on the 8 subcarriers for each user

packet; "message_type" presents the type of data in the packet; "packet_num" shows the number of the current packet; "packet_len" shows the size of the packet. Only if the list of information is printed in the console does the base station obtains the corresponding packet. In the functionality check for both uplink and downlink, we see most of the lists of information, which means the most of the packets reach the receiver.

Subsequently, we test the communication logic for both user device and base station. The criterion is the packet transmission and reception follows the communication logic mentioned in Section 4.1. We check the logic through the lists of information. The

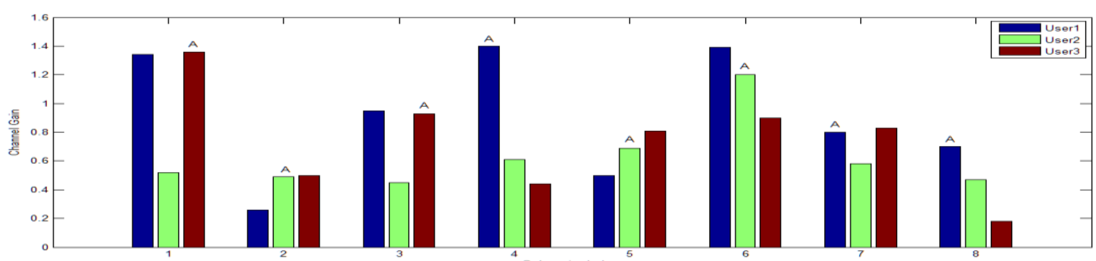


(a) Results obtained by GNU Radio

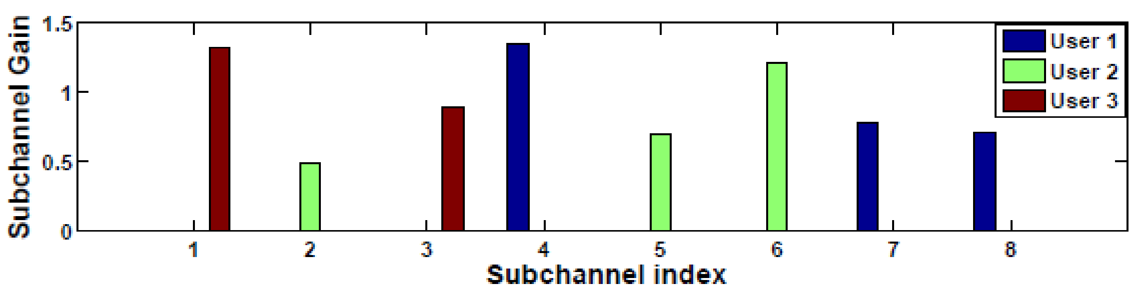


(b) Results obtained by [14]

Figure 5. 4 Result comparison between GNU Radio and [14] for Max-Sum algorithm

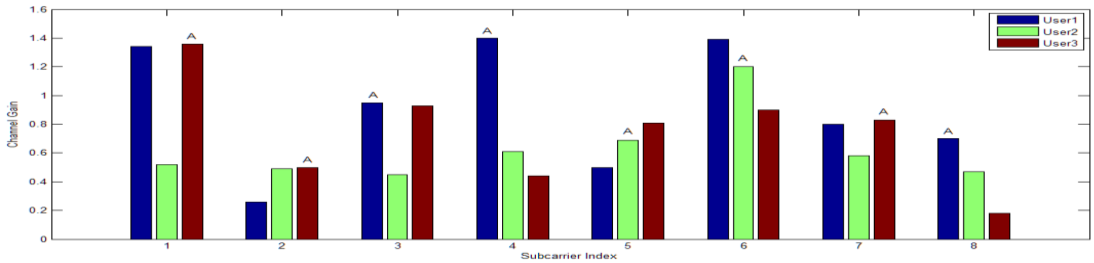


(a) Results obtained by GNU Radio

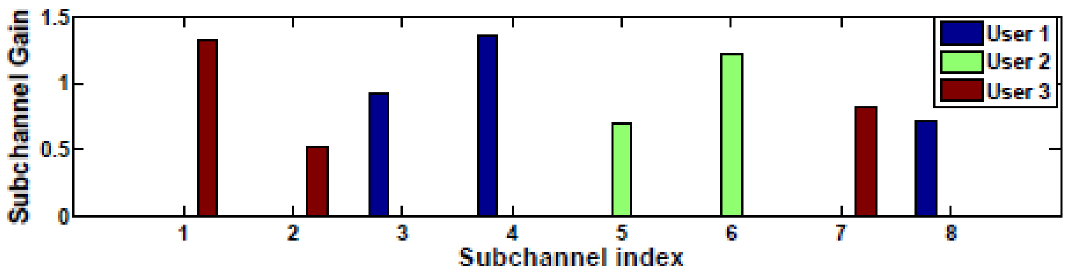


(b) Results obtained by [14]

Figure 5. 5 Result comparison between GNU Radio and [14] for Max-Min algorithm



(a) Results obtained by GNU Radio



(b) Results obtained by [14]

Figure 5. 6 Result comparison between GNU Radio and [14] for Max-Product algorithm

device id and message type are the critical information for logic test. At the perspective of the user device, the result is that after sending the request with device ID "12345" and message type "0", it receives the packet with the same device ID and message type "0", which means the user device receives the "acknowledgement btu" from base station. Then, the user device receives packets with the same device ID and message type "1" after sending acknowledgement to base station, which means the user device is receiving the requested data. Several seconds later, the packet with same device ID and message type "2" is received. Finally the user device sends a packet with message type "2" to base station and the session is over.

In conclusion, this series of behavior implies the functionality of the system works perfectly. We do not need to check the functionality from the other side because the entire logic binds each part of the system. If one works well, the others work too. However, we still checked it at the base station side and it works perfectly.

5.2.2 Dynamic Resource Allocation Algorithms Test

Before simulating with the channel models provided by GNU Radio, we test the algorithms based on the channel gains almost same as the simulation done by Akram Baharlouei [14] and compare the results obtained from GNU Radio to that in [14]. The channel gains for each user in [14] are shown in Figure 5.3.

Figure 5.4 shows the result of subcarrier allocation from Max-Sum algorithm. Figure 5.5 shows the result of subcarrier allocation from Max-Min algorithm. Figure 5.6 shows the result of subcarrier allocation from Max-Product algorithm. The "a" part of the three figures is the result obtained by GNU Radio, while the "b" part is the result from [14]. In "a" part of each figure, the bar marked by "A" means that this user occupies the corresponding subcarrier, while in "b" part the shown bar indicates that this subcarrier is allocated to this user. Comparing to the results of the three algorithms, the implementation of the three algorithms works perfect since all of the three algorithms achieve the same result as that in [14]. Thus, the correctness of the implementation is proved to be true by the result comparison.

5.2.3 Algorithms Simulation with GNU Radio Channel Model

The channel information is obtained by the user device based on the acknowledgement from the base station. Then, the information is sent back to base station along with the acknowledgement from the user device. We implemented Max-Sum, Max-Min and Max-Product algorithm respectively.

Figure 5.7 shows the result of subcarrier allocation from Max-Sum algorithm. The channel information of three user devices is plotted in a group bar chart. The bars which are marked with an "A" above indicate the subcarrier is allocated to the corresponding user. We see the channel condition of user 1 is the worst since no channel gain is better than both user 2 and 3. Furthermore, most of the channel gains are worse than the other two users. The gains of user 2 and 3 are higher but that of user 3 is more even. According to the Figure 5.7, the result of resource allocation from GNU Radio is exactly matches the analytical result by Max-Sum algorithm. All of the subcarriers are allocated to the user with best channel gain. Unfortunately, the user 1 lost all opportunities to obtain a channel in the 8 subcarriers. In conclusion, the largest rate of this algorithm is guaranteed but sometimes the user may have significant latency or lost connection from the communication.

Figure 5.8 shows a different solution for the resource allocation using Max-Min algorithm. Obviously, user 1, the worst user, obtains opportunities to utilize the subcarrier resources. According to the algorithm, each user obtains the subcarrier at the position of best channel gain. Then, the rest of the algorithm calculates the accumulation of the channel gains where the subcarriers are occupied and determine to allocate subcarrier to which user. Finally, we see total rate of each user has no large gap between each other.

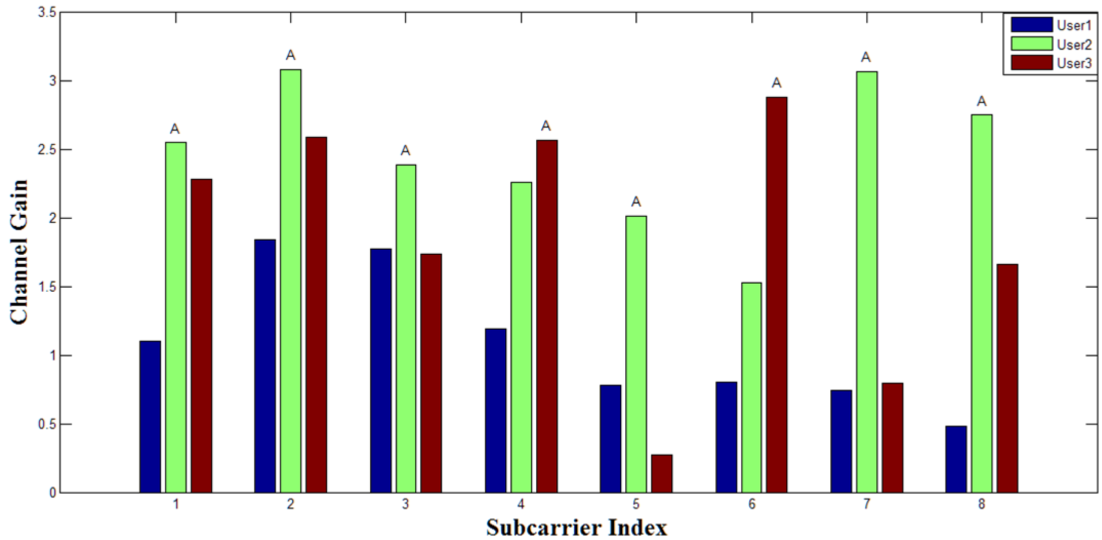


Figure 5. 7 Subcarrier allocation for Max-Sum algorithm

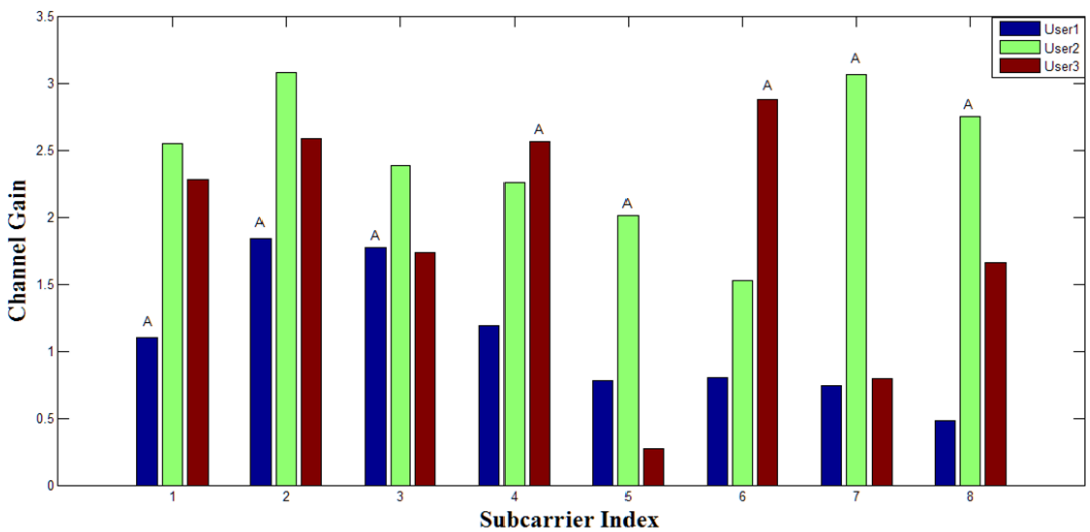


Figure 5. 8 Subcarrier allocation for Max-Min algorithm

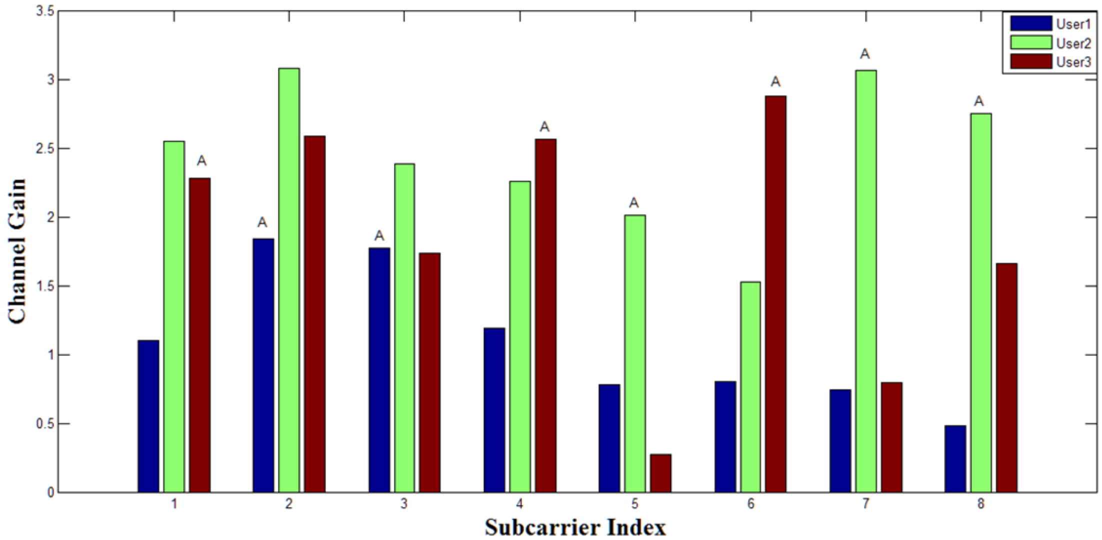


Figure 5. 9 Subcarrier allocation for Max-Product algorithm

Figure 5.9 shows the result of subcarrier allocation from Max-Product algorithm. We set $R_{i,\min}$ as 0 so that the first two stages of the algorithm are the same as Max-Min algorithm. The only difference is the determination condition in the third stage. Under the effect, we can see the difference resource allocation strategy on subcarrier 1, which is allocated to user 3. Since the channel gain of user 3 is larger than user 1, the data rate on subcarrier 1 by Max-Product algorithm is larger than that by Max-Min algorithm according to Shannon's Equation. The channel gains on other subcarriers keep same as the result of Max-Min algorithm. Eventually, this solution improves the total rate of the system comparing to Max-Min algorithm.

In conclusion, the result collected from GNU Radio matches the theoretical analysis of these dynamic resource allocation algorithms. Max-Sum only allocates subcarrier to the best channel gain; Max-Min allocate more subcarriers to the worst channel gain; Max-Product acts as a centrist which makes a moderate strategy.

Since the power allocation and the rate calculator are not completed using GNU Radio, the power allocation on each subcarrier is the same.

CHAPTER 6: CONCLUSION AND FUTURE WORK

This thesis supposes to implement the theory through real world. Three basic dynamic resource allocation algorithms are implemented: Max-Sum algorithm aims at maximizing the total rate of the communication system; Max-Min algorithm focuses on the fairness of dynamic resource allocation making sure the user with the worst channel condition has fair opportunities to obtain the communication resource; Max-Product algorithm introduces Nash Bargaining Solution to overcome the disadvantages of Max-Sum and Max-Min algorithms.

To implement these algorithms in the real world channel, we used GNU Radio, a highly flexible and low cost platform. Unlike the simulation on computer, GNU Radio can work with USRP for signal transmission and reception. To implement the dynamic resource allocation algorithms for LTE system, the thesis completes building the downlink and uplink system based on LTE technologies. The downlink and uplink work perfect in GNU Radio. Besides, the implementation of the behavior simulation of user device and base station is completed and work well in GNU radio.

The original plan for the thesis was to get the system working both with GNU Radio and its hardware device USRP. However, the limited time and high burden work of programming on GNU Radio left this part of work behind. Thus, the future work on the implementation of dynamic resource allocation for LTE system by GNU Radio is to let the user device and base station transmit and receive signal through USRP. Meanwhile, the base station is able to detect and establish connection with more than one user devices. Moreover, the power allocator will be built before transmission. Furthermore, the ultimate goal of this work is to implement an LTE base station on one computer and URSP so that our mobile phones can find the network provided by GNU Radio and send a call and message to each other, like the OpenBTS does.

APPENDIX A: RESOURCE ALLOCATION ALGORITHM SOURCE CODE

The three dynamic resource allocation algorithms consist of three main C++ functions for the implementation of the algorithms and four utility functions for finding the maximum or minimum value and corresponding index from a vector or an array.

A.1 Max-Sum Algorithm

```
vector<vector<int> > maxsum(vector<vector<double> > chan){
	vector<vector<int>> result;
	for (int i = 0; i < chan.size(); i++){
		vector<int> a;
		result.push_back(a);
	}

	double* temp = new double[chan.size()];

	for (int i = 0; i < chan[0].size(); i++){
		for (int j = 0; j < chan.size(); j++){
			temp[j] = chan[j][i];
		}
		result[find_max_index(temp)].push_back(i);
	}
	return result;
}
```

A.2 Max-Min Algorithm

```
vector<vector<int> > maxmin(vector<vector<double> > chan){
	//initialization
	vector<double> R(chan.size(),0);

	vector<vector<int> > result;
	for (int i = 0; i < chan.size(); i++){
		vector<int> a;
		result.push_back(a);
	}

	//step 2
	for (int i = 0; i < chan.size(); i++){
		int n = find_max_index(chan[i]);
		result[i].push_back(n);
```

```
        R[i] += chan[i][n];
        for (int j = 0; j < chan.size(); j++){
            chan[j][n] = 0;
        }
    }

    //step3
    while (find_max(chan[0]) != 0){
        int user = find_min_index(R);
        int n = find_max_index(chan[user]);
        result[user].push_back(n);
        R[user] += chan[user][n];
        for (int j = 0; j < chan.size(); j++){
            chan[j][n] = 0;
        }
    }
    return result;
}
```

A.3 Max-Product Algorithm

```
vector<vector<int> > maxproduct(vector<vector<double> > chan){
    //initialization
    double rmin = 0; // assume Rmin is 0 for all users

    vector<double> R(chan.size(), 0);
    vector<vector<double> > nbs_metric;
    vector<double> gain_allocated(chan.size(), 0);
    for (int i = 0; i < chan.size(); i++){
        vector<double> a;
        nbs_metric.push_back(a);
    }

    vector<double> user_set; // should be int
    for (double i = 0; i < chan.size(); i++){
        user_set.push_back(i);
    }

    vector<vector<int> > result;
    for (int i = 0; i < chan.size(); i++){
        vector<int> a;
        result.push_back(a);
    }

    //step 2
    //find user with largest gain
    while (find_max(user_set) >= 0){
        vector<double> temp;

        for (int j = 0; j < user_set.size(); j++){
            if (user_set[j] != -1){
                temp.push_back(find_max(chan[j]));
```

```
				}
				else{
					temp.push_back(0);
				}
			}
			int user = find_max_index(temp);

			while (R[user] <= rmin){
				int n = find_max_index(chan[user]);
				result[user].push_back(n);
				R[user] += chan[user][n];
				gain_allocated[user] += chan[user][n];
				for (int m = 0; m < chan.size(); m++){
					chan[m][n] = 0;
				}
				for (int k = 0; k < nbs_metric.size(); k++){
					if (nbs_metric[k].size() != 0){
						nbs_metric[k][n] = 0;
					}
				}
			}
			user_set[user] = -1;

			//calculate nbs metric
			for (int a = 0; a < chan[0].size(); a++){
				nbs_metric[user].push_back(chan[user][a] /
gain_allocated[user]);
			}

		}

		//step3
		while (find_max(chan[0]) != 0){
			vector<double> temp;

			for (int j = 0; j < nbs_metric.size(); j++){
				temp.push_back(find_max(nbs_metric[j]));
			}
			int user = find_max_index(temp);

			int n = find_max_index(chan[user]);
			result[user].push_back(n);
			R[user] += chan[user][n];

			//update nbs metric
			gain_allocated[user] += chan[user][n];

			for (int a = 0; a < chan[0].size(); a++){
				nbs_metric[user][a] = chan[user][a] / gain_allocated[user];
			}

			for (int m = 0; m < chan.size(); m++){
```

```
				chan[m][n] = 0;
			}
			for (int k = 0; k < nbs_metric.size(); k++){
				nbs_metric[k][n] = 0;
			}
		}
		return result;
}
```

A.4 Finding Maximum Value From Vector

```
double find_max(vector<double> a){
		double max_value = -1;
		for (int i = 0; i < a.size(); i++){
			if (a[i] >= max_value){
				max_value = a[i];
			}
		}
		return max_value;
}
```

A.5 Finding the Index of Maximum Value From Array

```
int find_max_index(double a[]){
		double max_value = 0;
		int max_index = -1;
		int l = sizeof(a);
		for (int i = 0; i < sizeof(a); i++){
			if (a[i] >= max_value){
				max_value = a[i];
				max_index = i;
			}
		}
		return max_index;
}
```

A.6 Finding the Index of Maximum Value From Vector

```
int find_max_index(vector<double> a){
		double max_value = 0;
		int max_index = -1;
		for (int i = 0; i < a.size(); i++){
			if (a[i] >= max_value){
				max_value = a[i];
				max_index = i;
			}
		}
		return max_index;
}
```

A.7 Finding the Index of Minimum Value From Vector

```
int find_min_index(vector<double> a){
	double min_value = 10000;
	int min_index = -1;
	for (int i = 0; i < a.size(); i++){
		if (a[i] <= min_value){
			min_value = a[i];
			min_index = i;
		}
	}
	return min_index;
}
```

REFERENCES

[1] A. Baharlouei and B. Jabbari, "Efficient and Fair Power and Subchannel Allocation in Multiuser OFDM Networks," *IEEE communications letters*, 17 (10), p. 1905 - 1907, 2013

[2] H. G. Myung, J. Lim, and D. J. Goodman, "Single Carrier FDMA for Uplink Wireless Transmission," *IEEE Vehicular Technology Magazine*, vol. 1, no. 3, pp. 30-38, Sep. 2006.

[3] H. G. Myung, "Introduction to Single Carrier FDMA," 15th European Signal Processing Conference, pp. 2144-2148, Sep. 2007.

[4] T.M. Schmidl and D.C. Cox, "Robust frequency and timing synchronization for OFDM," IEEE Transactions on Communications, 1997.

[5] G. Ren, H. Zhang and Y. Chang, "SNR estimation algorithm based on the preamble for OFDM systems in frequency selective channels," IEEE Transactions on Communications, VOL. 57, NO.8, August 2009

[6] W. Rhee, and J. Cioffi, "Increase in Capacity of Multiuser OFDM System using Dynamic Subchannel Allocation," in *Proc. IEEE VTC*, 2000.

[7] Z. Shen, J. G. Andrews and B. L. Evans, "Adaptive Resource Allocation in Multiuser OFDM Systems with Proportional Rate Constraints," *IEEE Trans. Wireless Commun.,* vol. 4, no. 6, 2005.

[8] J. Jang and K. B. Lee, "Transmit power adaptation for multiuser OFDM systems," *IEEE J. Select. Areas Commun*., vol. 21, no. 2, 2003.

[9] B. Mathias, "Dynamic resource allocation in OFDM systems: an overview of cross-layer optimization principles and techniques," *IEEE network*, 21 (1), p. 53 - 59, 2007

[10] N. Damji, "Dynamic Downlink OFDM Resource Allocation With Interference Mitigation and Macro Diversity for Multimedia Services in Wireless Cellular Systems," *IEEE transactions on vehicular technology*, 55 (5), p. 1555 - 1564, 2006

[11] T. Wu, "Dynamic and fair resource allocation algorithm for OFDM systems," *IEEE communications letters*, 11 (12), p. 931 - 933, 2007

[12] K. Letaief, "Advances in smart antennas - Dynamic multiuser resource allocation and adaptation for wireless systems," *IEEE wireless communications,*13 (4), p. 38 - 47, 2006

[13] Y. Zhang, "A Distributed Algorithm for Resource Allocation in OFDM Cognitive Radio Systems," *IEEE transactions on vehicular technology*, 60 (2), p. 546 - 554, Feb. 2011

[14] A. Baharlouei and B. Jabbari, "Dynamic Subchannel and Power Allocation using Nash Bargaining Game for Cognitive Radio Networks with Imperfect PU Activity Sensing," *8th International Conference on Cognitive Radio Oriented Wireless Networks*, 2013.

[15] 3GPP, "LTE; Evolved Universal Terrestrial Radio Access; Physical Channels and Modulation", 3GPP TS 36.211, Jul. 2013

[16] 3GPP, "LTE; Evolved Universal Terrestrial Radio Access; Multiplexing and Channel Coding", 3GPP TS 36.212, Jul. 2013

[17] 3GPP, "LTE; Evolved Universal Terrestrial Radio Access; Physical Layer Procedure", 3GPP TS 36.213, Jul. 2013

[18] 3GPP, "LTE; Evolved Universal Terrestrial Radio Access and Evolved Universal Terrestrial Radio Access Network; Overall Description", 3GPP TS 36.300, Sep. 2013

[19] 3GPP, "LTE; Evolved Universal Terrestrial Radio Access; Medium Access Control Protocol Specification", 3GPP TS 36.321, Jul. 2013

[20] W. Zhang, "Game and Society," Peking University Press, Jan. 2013

[21] H. Cheng, Thesis Source Code, https://github.com/harrychk/github-repository, last retrieve: Dec. 2 2014

[22] T. M. Cover and J. A. Thomas, "Elements of Information Theory," New York: Wile, 1991.

BIOGRAPHY

Hanke Cheng received his BS degree in Electrical Engineering from Wuhan University of Technology, China in 2011. Subsequently, he continued his MS degree in Electrical Engineering at George Mason University, Fairfax, VA in Spring 2011. Before the MS program, he completed the study at Linnaeus University, Sweden in 2011and got the BS degree in Computer Science. His research interests are GNU Radio, wireless communication.