THE COOPER UNION
ALBERT NERKEN SCHOOL OF ENGINEERING

Distributed Synchronization for Ad-Hoc Operation in LTE

by

David Li

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Engineering

Advised by: Sam Keene

THE COOPER UNION FOR THE ADVANCEMENT OF SCIENCE AND ART

THE COOPER UNION

This thesis was prepared under the direction of the Candidate's Thesis Advisor and has received approval. It was submitted to the Dean of the School of Engineering and the full Faculty, and was approved as partial fulfillment of the requirements for the degree of Master of Engineering.

Dr. Richard J. Stock - Sept. 12 2016

Dean, School of Engineering

Dr. Sam M. Keene - Sept. 12 2016

Candidate's Thesis Advisor

Acknowledgments

Firstly, I would like to thank my advisor, Professor Sam Keene, for putting up with my shenanigans during the course of this thesis.

I would also like to thank Daniel Cho for help with LaTeX formatting and Jason Tam for setting the bar.

Abstract

This thesis presents a new method of physical layer synchronization for wireless communications. In particular, LTE release 12 introduces new device to device capability that lays down some of the groundwork for incorporating an ad-hoc protocol into LTE itself. Our method consists of expanding upon existing synchronization signals already present in LTE such that physical synchronization can occur in a distributed multi-point to multi-point fashion as opposed to the typical point to point fashion. This allows for a greater amount of synchronization information to be passed between nodes in the system. In addition, we introduce aspects of a novel link layer design that utilizes our physical layer distributed synchronization method demonstrating the validity of the physical layer technique.

Contents

List of figures

Nomenclature

CSMA Carrier Sense Multiple Access

D2D Device to Device

eNodeB LTE base station

FDD Frequency Division Duplex

HARQ Hybrid Automatic Repeat Request

LTE Long Term Evolution

OFDMA Orthogonal Frequency Division Multiple Access

PAPR Peak to Average Power Ratio

ProSe Proximity Services

PSS Primary Synchronization Signal

PSSS Primary Sidelink Synchronization Signal

RSRP Reference Signal Received Power

SC-FDMA Single Carrier Frequency Division Multiple Access

SL Sidelink

SSS Secondary Synchronization Signal

SSSS Secondary Sidelink Synchronization Signal

TDD Time Division Duplex

UE User Equipment

Chapter 1

Introduction

The dominant wireless communications protocols currently being used by the average consumer currently depend on a star topology. The end devices are connected to a central hub through which all traffic is routed. This is the case with traditional cellular architecture where user equipment(UE) sends data to a base station which eventually gets routed to an end device or another UE.

Long Term Evolution (LTE) is the current generation of the commercial cellular standard [5]. Beginning with release 12, LTE is starting to standardize features that will eventually allow devices to communicate to other devices without first sending data to the base station (eNodeB), .

The initial seedling for these features came from Qualcomm in the form of a technology called FlashLinQ in 2010 [22]. FlashLinQ is a synchronous distributed scheduler for device to device operation. Qualcomm was able to demonstrate

prototype radios with some success hence motivating a push for incorporation of some of the concepts and ideas of FlashLinQ into discussion for future LTE releases.

Currently, the device to device (D2D) features are being marketed and published under several names including LTE Direct and LTE Proximity Services (ProSe).

As of March 2015, Release 12 of LTE has been frozen by 3GPP. As such, it is merely a matter of time before device manufacturers and network operators begin to build devices and infrastructure that will allow for the use of these D2D features.

One of the key challenges in any communications system is synchronization. In LTE, frame synchronization and carrier frequency synchronization are of particular interest. One of first goals of ProSe is to develop a synchronization method adequate for the initial constrained use scenarios for device to device.

1.1 Organization

We begin by introducing the concept of generic synchronization in OFDM systems which includes LTE in Chapter 2. We then begin to discuss some minimal background on the LTE physical layer including the mathematics behind some of the sequences used to generate synchronization signals in Chapter 3. This is followed by a discussion on the current state of ProSe in Chapter 4. Afterwards, our synchronization technique is discussed from a physical layer perspective in Chapter 5. Finally, the framework for a link layer design is introduced in Chapter 6.

Chapter 2

Background

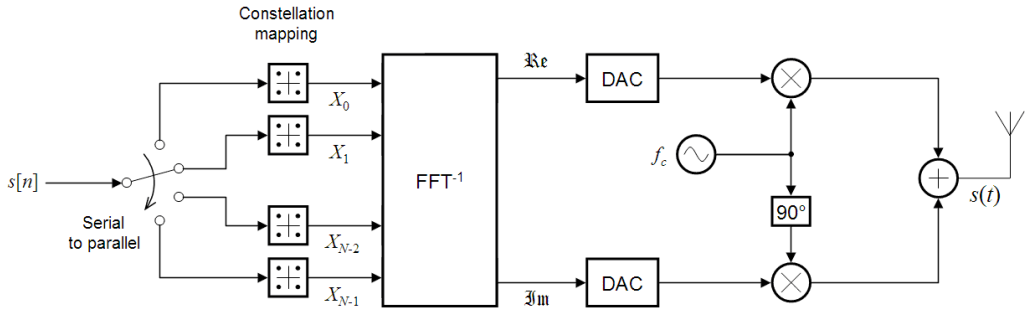
A typical digital communications system works by mapping a bit sequence to a symbol which corresponds to an analog waveform at baseband [13]. The transmitter then mixes the baseband waveform up to some higher radio frequency for transmission over an analog channel. At the receiver, the signal is mixed back down and processed at baseband.

2.1 Synchronization in OFDM systems

Orthogonal Frequency Division Multiplexing (OFDM) is a digital modulation method that relies on a number of orthogonal subcarrier signals to carry data over the channel. OFDM has become one of the more popular modulation methods mostly due to its high spectral efficiency and ease of implementing equalizers. It is currently being used in several commercial waveforms including digital subscriber

line (DSL), the 802.11 standards (commonly known as WiFi), DVB-T, and LTE [12],[6],[9],[5]. OFDM relies heavily on the receiver having its carrier frequency, sampling frequency, and symbol times synchronized in order to operate at reasonable performance.

Figure 2.1.1: Ideal OFDM Transmitter[21]



As seen from Figure 2.1.1, OFDM is a block based method where a bit stream is buffered, and then operated upon as a block passing through the IFFT operation. Then, it is serialized again for the actual transmission over the air, $s(t)$. Each one of the symbols in an IFFT bin corresponds to a subcarrier. A cyclic prefix (not shown in Figure 2.1.1) is usually added after the IFFT step. It serves to act both as a separator between OFDM symbols and so the channel can be modeled as operating on the waveform with circular instead of linear convolution, hence allowing for easy equalizer implementation.

An arbitrary carrier frequency offset can broken down into two separate offset components. The first is an integer multiple of the subcarrier spacing. For an integer offset, the incorrect constellation symbols will be decoded off the subcarriers resulting in correct bits but in the incorrect position within the bit stream (along

with some incorrect bits from the newly introduced constellation symbols on subcarriers that weren't being used at the transmitter end). A fractional subcarrier frequency offset results in inter carrier interference (ICI) between the previously orthogonal sub-carriers resulting in severely degraded performance. Hence, frequency synchronization is desired.

Symbol level synchronization is needed since OFDM is a block based method. The serial stream being processed needs to correspond to an original OFDM symbol as opposed to having time samples from two different OFDM symbols. Without symbol level synchronization, there will be both inter symbol and inter carrier interference. The cyclic prefix can absorb some offset in the symbol level synchronization as well. However, it is typically fixed to a small size since it operates as overhead. The cyclic prefix only needs to be long enough to account for the delay spread of the multipath channel, but any extra length can be used to absorb sample offsets in the symbol level synchronization since they can be easily handled during equalization.

In LTE, the standard specifies that the carrier clock and the sample clock are both derived from the same original local oscillator [3]. Suppose the carrier clock is 1 GHz and the sample clock is 1 MHz. If the sample clock is 1 Hz slow, that means that the carrier clock is 1 kHz slow. LTE systems typically analyze and adjust for carrier frequency offset since that will also correct the sample clock. Sample clock offset typically has minimal impact on the LTE waveform compared to carrier frequency offset. The sample clock usually runs at approximately two to three orders of magnitude slower than the carrier clock. This, combined with the typical

OFDM symbol sizes in LTE, means that if the sample clock offset is significant over one or two OFDM symbols, the carrier frequency offset would be in the range of 10 MHz. For context, the maximum bandwidth mode of LTE is only 20 MHz, so this offset is huge. Hence, it is sufficient to be concerned only with carrier frequency and OFDM symbol level synchronization in LTE.

Chapter 3

Traditional LTE

The framing structure and synchronization procedures used in LTE will provide the necessary background in order to understand the synchronization procedures used in ProSe and ultimately, the proposed procedure.

3.1 Framing

LTE utilizes orthogonal frequency division multiple access (OFDMA) on the downlink and single carrier frequency division multiple access (SC-FDMA) on the uplink [13]. OFDMA is just OFDM with the logical allocation of sets of subcarriers to different users. SC-FDMA is just OFDMA with a discrete Fourier transform precoding step in the OFDMA transmit chain.

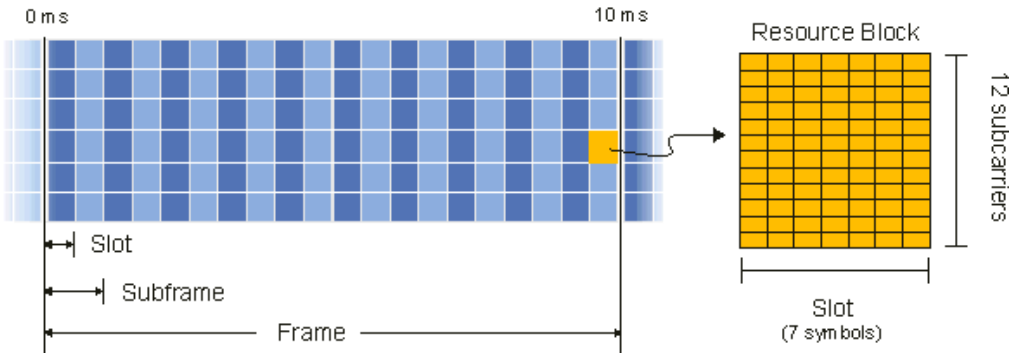
We can organize LTE resources in a time frequency grid, which at its most granular is OFDM symbols in time by subcarriers in frequency. For the purposes of

this discussion, we can ignore the dimension of space which is introduced by using multiple antennas. In addition, we also focus on the normal cyclic prefix mode framing structure as opposed to the extended cyclic prefix mode which uses a longer cyclic prefix on the OFDM symbols.

LTE has two modes of operation, Time Division Duplex (TDD) and Frequency Division Duplex (FDD). In TDD mode, subframes are designated as uplink, downlink, or special. Special frames are a hybrid of uplink, downlink, along with a period of dead time that is used for switching between the two. In FDD mode, there is a designated bandwidth at a different carrier frequency for uplink and downlink operations. The standard supports both FDD and TDD mode, although FDD mode seems to be dominant. Of the major carriers in the United States, only Sprint supports a single LTE band that operates in TDD mode. In this work, we focus exclusively on FDD mode.

The major unit of time in LTE is a 10 ms unit called a frame. These are divided into 1 ms subframes. Each subframe is further divided into 0.5 ms slots. Each slot is 7 OFDM symbols long in time. In frequency, each subcarrier is occupies 15 kHz of bandwidth, with varying numbers of subcarriers for the different bandwidth modes. A 12 subcarrier by 7 OFDM symbol component is referred to as a resource block. Resource blocks are further divided into 84 resource elements which are 1 subcarrier by 1 OFDM symbol.

Figure 3.1.1: LTE FDD Frame for 1.4 MHz with Normal Cyclic Prefix[20]



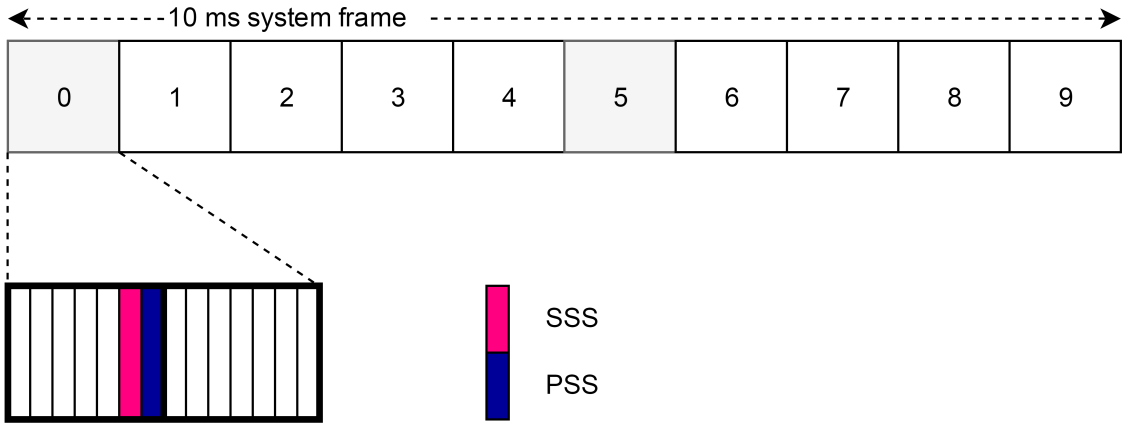
3.2 Synchronization

When a UE wishes to establish a connection, it must first go into cell search mode. Cell search consists of searching for and attempting to synchronize to the LTE synchronization signals being broadcast by a nearby cell. LTE provides two reference signals on the downlink channel for synchronization purposes. These are the Primary Synchronization Signal (PSS) and the Secondary Synchronization Signal (SSS) . Time alignment, sampling clock synchronization, and carrier frequency synchronization are the three main synchronization goals. Using only one sequence is sufficient to obtain symbol and sample synchronization. However, frequency synchronization can only be reliably obtained by using more time samples hence the addition of a secondary sequence. These sequences also provide ancillary system information including the Physical Cell ID, cyclic prefix mode, and whether the operation is in FDD or TDD mode. This provides some of the physical layer parameters that are needed to proceed with the connection procedure.

Within the frame, the PSS is located in the last OFDM symbol during the 1st and

11^{th} slots. The SSS is located in the adjacent OFDM symbol in time directly before the PSS as shown in Figure. 3.2.1. Both signals use the middle 6 frequency RBs of the LTE resource grid. This means that the synchronization signals for all bandwidth modes can be processed at a low base sample rate that is independent of the bandwidth mode the LTE cell is operating at.

Figure 3.2.1: PSS & SSS in LTE Frame



3.2.1 Primary Synchronization Sequence

The PSS is constructed using a Zadoff-Chu sequence [14]. The formulation for this family of sequences is shown in 3.1

$$A_q(n) = exp\left[-j2\pi q\frac{n(n+1)/2+ln}{N_{ZC}}\right] \qquad \begin{array}{l} 1 \leq q \leq N_{ZC}-1 \\ 0 \leq n \leq N_{ZC}-1 \\ l \in \mathbb{N} \end{array} \tag{3.1}$$

where q is refered to as the root, and N_{ZC} is the length of the sequence. In the case of LTE, l is always 0 for ease of implementation.

A Zadoff-Chu sequence is a complex sequence of constant magnitude which possesses some desirable properties. One property is that the N_{ZC} point DFT is also constant magnitude. An issue with using OFDM is that the waveform has a higher Peak to Average Power Ratio (PAPR) than many other waveforms [13]. This results in the need to use analog-digital convertors with a higher dynamic range and power amplifiers that need to be linear over a larger range.

Zadoff-Chu sequences have a zero cross correlation property. Circularly shifted versions of a Zadoff-Chu sequence will have zero cross correlation. Hence, if given one Zadoff-Chu sequence and another that was generated by circularly shifting the first, one can compute the amount of shift by simply cross correlating and shifting one of the sequences until a non zero result occurs.

Zadoff-Chu sequences also satisfy the property shown in Equation 3.2

$$\frac{1}{N_{ZC}} = \left| \frac{1}{N} \sum_{n=0}^{N_{ZC}} a_{q_1}(n) b^*_{q_2}(n+l) \right| \qquad \begin{array}{l} 0 \leq l \leq N_{ZC} - 1 \\ |q_1 - q_2| \perp N_{ZC} \end{array} \tag{3.2}$$

For any two Zadoff-Chu sequences where the absolute value of the difference of the roots is coprime to the length, the cyclic cross correlation function is constant in magnitude.

In implementation, the PSS is generated by mapping a Zadoff-Chu sequence of length 63 to resource elements within the middle 6 resource blocks. The 32$^{\text{nd}}$ element doesn't get mapped to any resource element. LTE downlink has a DC subcarrier which is not transmitted on. In this case, this corresponds to one of the elements in the Zadoff Chu sequence. The other resource elements above and below the PSS resource elements in frequency are also unused and nulled. For PSS generation, LTE uses 3 root numbers: 25, 29, and 34 [2].

Detection of the PSS is typically done by a sample level search. The maximum likelihood estimator for the sample offset in time is given by Equation 3.3

$$m^*_M = \text{argmax}_m \left| \sum_{t=0}^{N-1} Y[t+m] S^*_M[t] \right|^2 \tag{3.3}$$

where t is the time index, m is the sample offset in time, and N refers to the

length of the reference PSS time domain signal . Since the PSS is transmitted every 5 ms, a UE can scan 5 ms of time samples for a PSS detect. Detection can be assisted by considering soft information over multiple 5 ms scans as well [17].

3.2.2 Secondary Synchronization Signal

The basis for the Secondary Synchronization Signal (SSS) are M-sequences [10]. These binary sequences are constructed by going through every possible state of a shift register. Two M-sequences of length 31 are modulated into BPSK symbols and are then scrambled and interleaved in the frequency domain to generate the a sequence of length 63 which again gets mapped to resource elements within the middle 6 resource blocks. Again, one of the elements of this sequence is ignored for the DC subcarrier and the other resource elements above and below in frequency within those 6 resource blocks are also nulled. For exact details on the scrambling and interleaving see [2].

Chapter 4

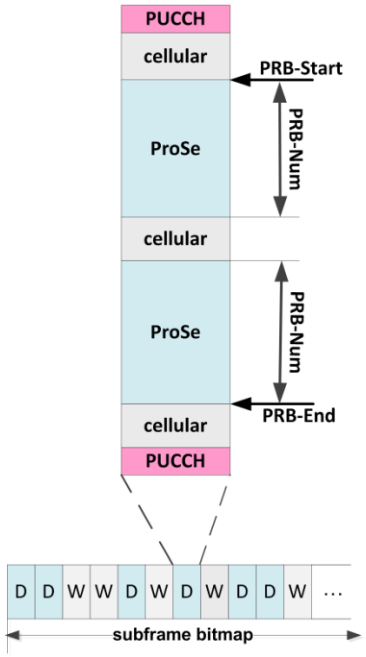
Proximity Services LTE

Proximity Services (ProSe) is a new introduction in 3GPP Release 12 that is intended to provide features that facilitate D2D communications. The three key features are D2D discovery, a D2D communications mechanism, and network level support D2D discovery and communications.

ProSe, as currently envisioned, is intended to supplement normal LTE operation. Resources are allocated to ProSe or Sidelink (SL) communications, as it is often referred to, at the beginning of each SL control period. This allocation is done via a subframe-resource block mask that is referred to as a Resource Pool as shown in Figure 4.0.1 The allocation is constrained given that some resources are always reserved to preserve normal cellular operation and by how they are specified using three parameters: PRB-Start, PRB-Num, PRB-End. Within the given resource allocation, there are resources allocated for channels that are analogous to the typical LTE channels for the purposes of broadcast and physical layer control

messaging.

Figure 4.0.1: Resource Pool Allocation for Sidelink [16]



4.1 Synchronization

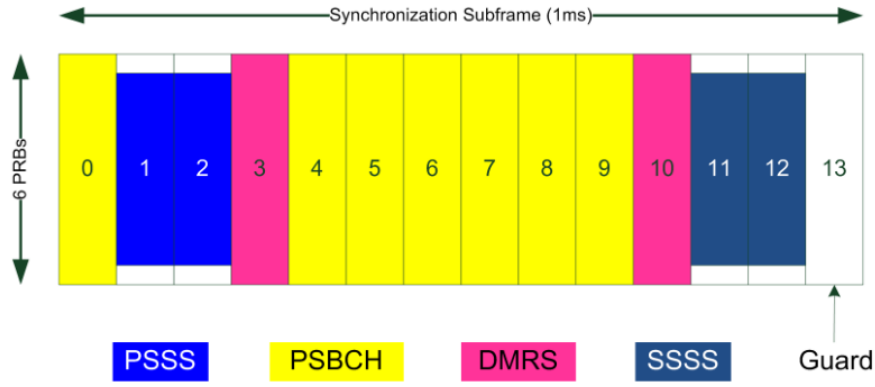
Synchronization in ProSe is accomplished by using primary and secondary synchronization signals known as Primary Sidelink Synchronization Signal (PSSS) and Secondary Sidelink Synchronization Signal (SSSS). These two signals share many similarities with the PSS and SSS respectively for normal LTE downlink operations.

4.1.1 Signal Construction

The PSSS and SSSS are constructed at baseband in almost the same fashion as the PSS and SSS. The PSSS and SSSS are mapped to the same subcarriers, but use SC-FDMA instead of OFDMA modulation. This choice is motivated by the PAPR issues which are typical of OFDM that led to the use of SC-FDMA for normal LTE uplink transmissions. Whereas the PSS and SSS occupy 1 OFDM symbol each, the PSSS and SSSS occupy 2 OFDM symbols each. Since the PSS is transmitted periodically, it is not critical for an UE to be able to detect and synchronize to it after the first instance it receives. In fact, if a UE is limited by processing power, it can choose to scan subsets of the search space since the transmissions are unchanging and occur at predictable intervals. Since the PSS and SSS are consistent for a given cell, a UE can perform detection and synchronization methods that use prior history for better performance. In ProSe however, the UE needs to be able to detect the PSSS and the SSSS within the first synchronization frame since the schedule for PSSS and SSSS transmissions is unknown to the UE and isn't periodic.

The PSSS and SSSS essentially consist of two copies of a modified PSS and a modified SSS respectively in a different framing structure. Having each signal span two OFDM symbols allows for symbol level search techniques as opposed to the sample level search that was the maximum likelihood estimator for the PSS shown prior in Equation 3.3. The extra OFDM symbol is essence acts as a large cyclic prefix.

Figure 4.1.1: Synchronization Subframe in ProSe [16]

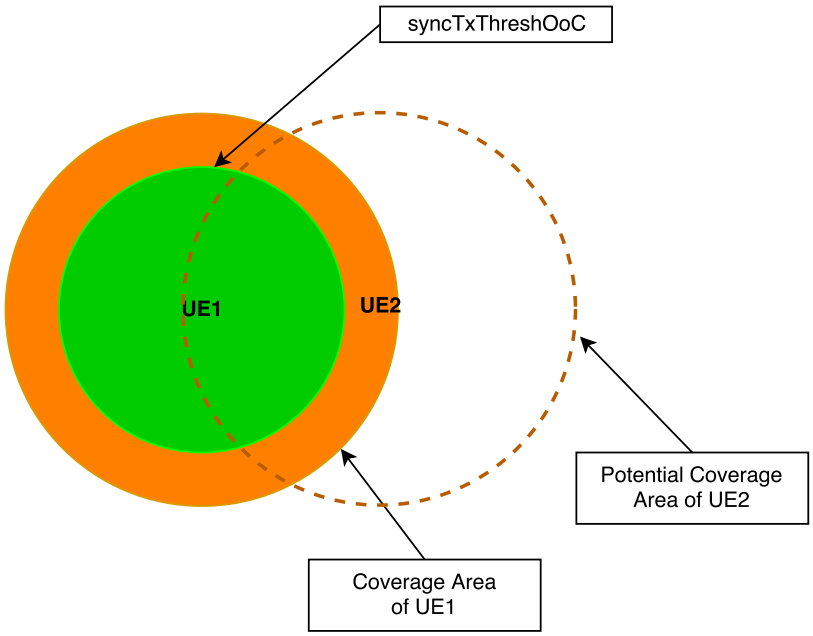


4.1.2 Transmission of Synchronization Signals

There are currently three scenarios for when a UE decides to transmit the PSSS/SSSS and then act as a mini LTE cell. First, the network could issue a RRCConnectionReconfiguration message that specifies the parameters with which a UE begins transmitting the PSSS and SSSS. This implies that the UE is in network range. In another case, the network could effectively cause UEs to begin transmitting the PSSS and SSSS by manipulating particular threshold parameters in the broadcast system information blocks. UEs can measure reference signal received power (RSRP) against the syncTxThreshIC level set by the network. If the RSRP is below the syncTxThreshIC threshold, then the UE has the opportunity to send the PSSS and SSSS whenever the higher layers direct it to use the sidelink channel. The last and most interesting case is for UEs which are out of network coverage. UEs out of coverage can search for PSSS and SSSS transmissions from three different categories of UEs. Those are UEs in network coverage, UEs synchronized to a UE in network coverage, and all other UEs. As synchronization sources, those groups of

UEs are prioritized in that order. Within the highest priority group, the highest RSRP is used to determine which UE to synchronize to. Then, it compares the RSRP of the selected synchronization source and chooses to transmit the PSSS and SSSS in another SL subframe if the RSRP is below a level set by syncTxThreshOoC, which is simply a preconfigured value provided by the higher layers [16].

Figure 4.1.2: Effect of threshold for decision on becoming a synchronization source



As shown in Figure 4.1.2 the effect of a threshold, whether it be one set by the network (for UEs in network coverage) or one set internally by the handset (for UEs in coverage of another UE), is to provide a mechanism to allow UEs that are on the cell periphery to transmit synchronization signals and in effect extend the coverage area of the network. If a UE decides to transmit the synchronization signals, it schedules the transmission of its synchronization signals to a subframe which isn't being used by its parent synchronization source.

4.2 Implications

The initial features of ProSe allow for D2D communications to occur within the context of LTE. As designed, ProSe requires at least an indirect connection to a network backbone. There is no procedural mechanism for UEs to act as a true ad-hoc network. As currently designed, certain UEs are designated as synchronization sources, i.e. they can operate on LTE downlink. To a UE not designated as a synchronization source, very little has actually changed besides some of the processing of the physical layer. The sidelink provides analogs for LTE downlink control which are essentially identical, but modified to fit the constraints of ProSe. Whereas traditional cell towers use an IP based backhaul over some other physical technology, ProSe allows UE synchronization sources to act in UE mode or sidelink synchronization source mode in order to emulate a backhaul for communication between synchronization sources. As such, ProSe synchronization is simply a reuse of the typical point to point synchronization mechanism.

Chapter 5

Physical Sync Source Detection

The original impetus for ProSe was the demonstration of a Qualcomm technology called Flashlinq. The core of this technology was to provide a mechanism where nodes could schedule traffic in a distributed fashion by use of a synchronous scheduler. The scheduler requires multiple nodes to simultaneously signal in time in order to schedule traffic for the network as a whole. Traffic operates without the use of an access point and goes directly from node to node. This is the motivation behind this exploration of distributed synchronization. We focus exclusively on establishing a framework focused on reusing the PSSS for the purposes of detecting a synchronization source.

5.1 PSSS Set Expansion

As ProSe is currently designed, a sidelink downlink subframe with a synchronization signal payload is designed to be used by a single user. This is clear from the resource allocation parameters as shown previously in Figure 4.0.1 since the synchronization payload lives inside the middle resource elements. In standard LTE, three Zadoff-Chu roots are used for PSS generation. In ProSe, an additional two Zadoff-Chu roots are introduced for use in the PSSS. We propose that multiple UEs can simultaneously broadcast synchronization signals. In order to facilitate this, the set of allowable PSSS roots needs to be expanded beyond two.

5.1.1 Alternative Detection Method

The maximum likelihood estimator for timing offset using the PSS was previously described in Equation 3.3. It is a known property of Zadoff Chu sequences that the DFT of a Zadoff Chu sequence is actually another Zadoff Chu sequence [7]. The exact relationship is shown in Equation 5.1.

$$A_q(n) = a_q^*(q^{-1}n)A_q[0] \tag{5.1}$$

Hence the estimator given by Equation 3.3 could still apply even though SC-FDMA modulation is used instead of OFDM. There is also another minor difference since SC-FDMA modulation in LTE does not have a null DC subcarrier like OFDM downlink. Instead, the DC exists at the boundary between the two

middle subcarriers.

However, the original maximum likelihood estimator relies on a sample level search which is undesirable from a computational point of view since the unpredictability of the sidelink synchronization subframe locations requires that immediate detection be made. Instead, we introduce another estimator which takes advantage of the repeated symbol and allows for an OFDM symbol level search.

$$s_I = \text{argmax}_{s_i} \left| \sum_{n=0}^{N-1} a(s_i T + n) b^*(n + l) \right|^2 \tag{5.2}$$

where N is the length of the reference PSSS signal b, s_i corresponds to an OFDM symbol index, and T corresponds to the length of an OFDM symbol in samples.

The cross correlation results from all pairwise combinations is shown in Table A.0.1. It is desirable to have a set of roots such that the max cross correlation peak is low. A high cross correlation peak could contribute to an incorrect detection. The estimator described in Equation 5.2, describes how to select the best reference PSSS as opposed to whether a particular PSSS is present.

To determine whether a particular PSSS is present, the metric we use is the ratio of the max value of the magnitude of the cross correlation peak to the average magnitude of the cross correlation function itself after taking out the peak itself. This particular metric is agnostic to overall power level variations. No matter how $a(n)$ and $b(n)$ are scaled, the max and mean also scale appropriately since those are linear operations. We can adjust this threshold in order to make trade offs between

missed detections and false alarms. A discussion of specific values used is in Section 5.2.4.

5.1.2 Impact of Interference with PSS

We also need to consider how the modified PSSS impacts PSS detection as well. There are scenarios where UEs are in range of a network it may still be advantageous to operate some D2D traffic for throughput reasons. Hence, we analyze potentially transmitting synchronization signals in subframes during a normal cell's synchronization subframes. There is also the possibility that the D2D network and a cellular network in range are uncooperative. As designed, ProSe allows an UE to coexist with its associated cellular network. However, there can be other cellular networks within network range on at the same frequency. The preference in these cases is to minimize the effect of the D2D network on the infrastructure network. In the uncooperative case, the D2D synchronization signals can interfere with other traffic as well. A combination of error correction coding and Hybrid Automatic Repeat Request (HARQ) can provide protection at the physical layer for D2D synchronization signals interfering with generic traffic of another cell. Higher layer mechanisms such as retransmission with TCP provide additional protection for IP based traffic as well. In the case of interference with the PSS, there is no protection mechanism and hence the scenario boils down to the same one prior where a synchronization source wishes to transmit the PSSS at the same time as a cell network's PSS transmission.

We can see based on Table 5.1.1 there are no major differences in the cross

correlation peaks between combination of possible roots for the PSSS and the current PSS roots. The conclusion from this table is that from the perspective of PSS detection, it doesn't matter which root sequence was used as the basis for PSSS construction.

Table 5.1.1: PSS - PSSS Cross Correlation

Root	**25**	**29**	**34**	**Root**	**25**	**29**	**34**
2	0.22	0.20	0.26	**32**	0.23	0.20	0.24
4	0.27	0.28	0.24	**34**	0.22	0.24	0.21
5	0.25	0.23	0.26	**37**	0.22	0.31	0.22
8	0.21	0.20	0.22	**38**	0.19	0.23	0.27
10	0.24	0.24	0.23	**40**	0.20	0.21	0.20
11	0.23	0.26	0.22	**41**	0.24	0.22	0.22
13	0.21	0.22	0.21	**43**	0.22	0.22	0.21
16	0.23	0.24	0.20	**44**	0.25	0.20	0.22
17	0.22	0.29	0.21	**46**	0.19	0.21	0.29
19	0.34	0.20	0.18	**47**	0.26	0.19	0.23
20	0.26	0.21	0.22	**50**	0.21	0.21	0.22
22	0.20	0.23	0.23	**52**	0.22	0.22	0.27
23	0.25	0.20	0.21	**53**	0.22	0.22	0.23
25	0.18	0.28	0.24	**55**	0.21	0.21	0.20
26	0.22	0.21	0.30	**58**	0.22	0.27	0.23
29	0.24	0.21	0.24	**59**	0.21	0.22	0.28
31	0.22	0.23	0.21	**61**	0.24	0.25	0.19

5.1.3 Power Considerations

Power considerations would also be something to consider in deciding the the set of PSSS indices. We look at cubic metric which is defined in Equation 5.3.

$$\text{CM} = \frac{20\log_{10}(\text{RMS}[v_{norm}^3(t)]) - 20\log_{10}(\text{RMS}[{v_{ref}}_{norm}^3(t)])}{K} \tag{5.3}$$

Cubic metric [15] along with PAPR/Crest Factor are popular metrics to assess the amount of power back off needed for a high powered amplifier. Cubic metric attempts to measure the effects of 3rd order non-linearities which cause adjacent channel leakage. Empirically, this has been shown to map well to the amount of power back off needed for the high power amplifiers typically used. The reference waveform v_{ref} along with empirical slope factor K are somewhat arbitrary as they are meaningful only as in relation to how much a power amplifier would need to back off with a reference waveform. In this case, we use reference values consistent with agenda item discussions at the 3GPP working group [11].

Table 5.1.2: Cubic Metric for PSSS

Root	**CM (dB)**	**Root**	**CM (dB)**
2	-0.30	**32**	1.00
4	0.01	**34**	1.21
5	-0.08	**37**	1.43
8	0.15	**38**	1.34
10	0.46	**40**	1.20
11	0.49	**41**	1.13
13	0.21	**43**	0.90
16	0.40	**44**	0.93
17	0.61	**46**	0.58
19	0.90	**47**	0.40
20	0.89	**50**	0.23
22	1.11	**52**	0.45
23	1.16	**53**	0.42
25	1.37	**55**	0.15
26	1.49	**58**	-0.04
29	1.19	**59**	-0.06
31	1.01	**61**	-0.27

As we can see from Table 5.1.2, we can see there is slightly less than 2 dB difference between the root with the highest cubic metric and the root with the

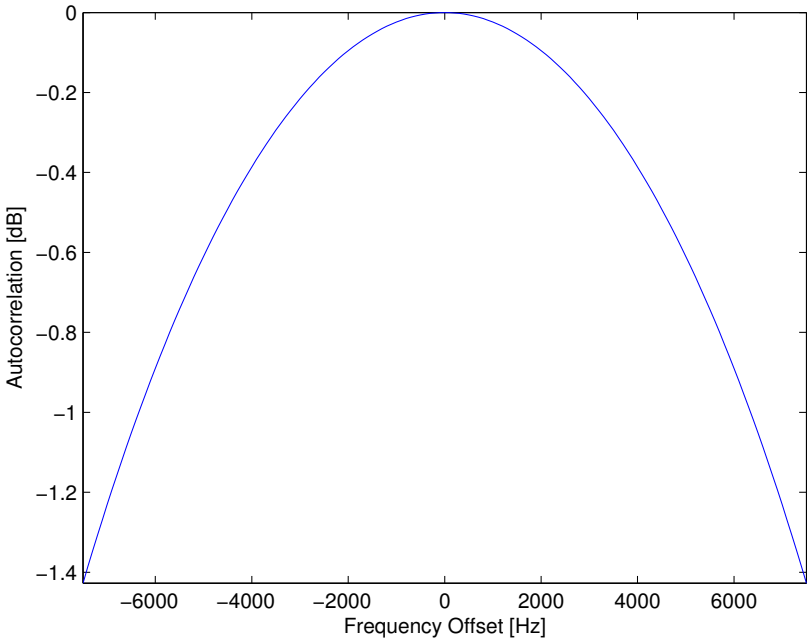
lowest cubic metric. It is interesting to note that the two roots that are currently used in the standard (26,37) for the PSSS actually have the highest cubic metric of all possible roots to choose from.

5.1.4 Carrier Frequency Offset Robustness

One of the trickier points of analysis is the consideration of performance in the presence of carrier frequency offset. An LTE band is typically larger than the operating bandwidth for a LTE network. For example, Band 1 of LTE is actually 60 MHz wide but the typical network operating on Band 1 can range from 1.4 MHz to 20 MHz in practice. The case we aim to address is an offset of at most 20 ppm which for a nominal frequency of 1 GHz is a 20 kHz offset. This is greater than a single LTE subcarrier which is only 15 kHz. For all PSSS roots, there is effectively no difference between the auto correlation peak strength for a given frequency offset. The effect of a carrier frequency offset on the PSSS autocorrelation peak strength is shown in Figure 5.1.1. For frequency offsets within half a subcarrier, there is only at most 1.4 dB of difference in peak strength. As such, we generate our reference hypotheses at half carrier spacing.

Given these considerations, it seems empirically a good choice of two additional roots to add to the set used would be 4 and 59. Roots which sum to the sequence length are conjugates of each other as can be shown by applying Equation 3.1. This conjugate symmetry is preserved in the further transformations with the DFT and IDFT. Conjugate root pairs are chosen since there exist methods which take

Figure 5.1.1: Average PSSS Autocorrelation Peak



advantage of this conjugate symmetry and simplify the computational complexity of detecting two sequences into that of one sequence [8]. This is consistent with the design philosophy for choosing the original PSS and PSSS roots where (29,34) and (26,37) are conjugate symmetric root sequence pairs.

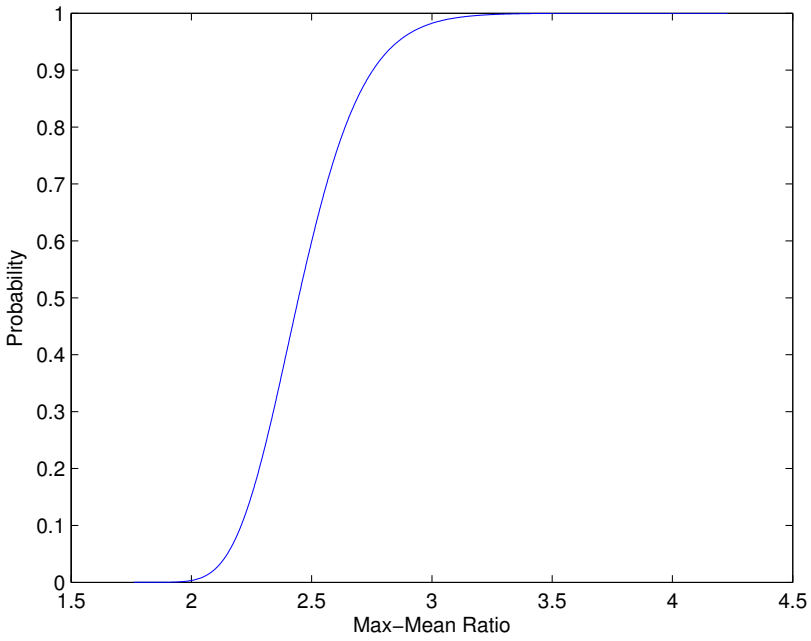
5.2 Probabilistic Model for Detection

We proposed to expand the set of root sequences to use for the PSSS in order to facilitate multiple UEs simultaneously transmitting synchronization signals. This allows propagation of synchronization information from several nodes at the same time as opposed to the current model of synchronization information propagating from one node.

5.2.1 In Presence of White Noise

We compute a baseline model for the statistics of the cyclic cross correlation function using the reference waveforms and white noise. Our detection metric is the ratio of the max magnitude of this function to the average magnitude.

Figure 5.2.1: CDF of Max-Mean Ratio of Cross Correlation Function w/ White Noise



We choose an empirically acceptable false alarm rate given the prior that synchronization signals nominally are transmitted on two OFDM symbols in a 14 OFDM symbol subframe. Hence the prior probability for a random OFDM symbol to contain a synchronization signal is 14.2% assuming all possible subframes do indeed contain a synchronization signal. Due to the repeated symbol for the synchronization signals, unless the OFDM sampled and the OFDM transmitted are exactly aligned, one OFDM symbol will contain a full copy of the synchronization

signal and the other two will contain parts of the synchronization signal. We can use the detection metric to determine between possibilities of multiple OFDM symbols register as detections. The penalty for a false alarm would be proceeding to SSSS detection which would very likely fail and hence not actively impact the UE's sense of the synchronization parameters. In addition, after initial acquisition, a UE will have some information about where the beginning of a subframe is and hence can employ priors on which OFDM symbols it expects the synchronization signals to exist on. The false alarm rate we choose is 2.5% which corresponds to a ratio of 2.95.

5.2.2 In Presence of Interference

As we can see in Table 5.2.1, all the max-mean ratios between our selected roots are below the average max-mean ratio when using white noise. We treat noise and interefernce as the same and choose to combine them into SINR. We use the white noise cross correlation statistics which are more aggressive than those of our selected roots.

Table 5.2.1: Max-Mean Ratio of Cross Correlation Function for References

Root	**4**	**26**	**37**	**59**
4	23.23	2.51	2.46	2.20
26	-	15.77	2.49	2.46
37	-	-	15.77	2.51
59	-	-	-	23.23

The motivation behind making this assumption goes back to Zadoff Chu sequences satisfying Equation 3.2. This equation says that for pairs of roots where the absolute difference is coprime to the sequence length, the value of the cross

correlation constant is a fixed constant, K.

$$K = \mathrm{E}\left(\sum_{n=0}^{N_{ZC}-1} w_0(n)w_1(n)\right) \tag{5.4}$$

where w refers to a white noise sequence with power equivalent to the PSSS sequence. This shows that the magnitude of the ideal cyclic cross correlation between different root pairs is the expected value of the magnitude of the cross correlation of two equal power white noise sequences. Since we are doing processing in the time domain using a Zadoff Chu sequence with a punctured element, this isn't the same scenario. However, the dualities between Zadoff-Chu in time and frequency provide some mathematical insight into why interference compared to white noise of the same power is less of a detriment. However, after we pass the PSSS signal through our fading channel model and sum a bunch of them, the new interference should look like white noise.

5.2.3 Propagation Model

The propagation model we use is ITU Pedestrian A [1] with 3 Hz of Doppler. This corresponds to approximately 3 $\mathrm{km\,h^{-1}}$. This Rayleigh fading channel is one of several standardized channel models used to assess LTE performance.

The other parameters relevant to generating the probabilistic model are shown in Table 5.2.3.

Table 5.2.2: ITU Pedestrian A

Delay [ns]	**Gain [dB]**
0	0
110	-9.7
190	-19.2
410	-22.8

Table 5.2.3: Model Generation Parameters

Nominal Freq	1 GHz
Fading Model	ITU-Pedestrian A
PPM offset	[-20, 20]
SINR	[2, - 16]
Iterations Per	10000

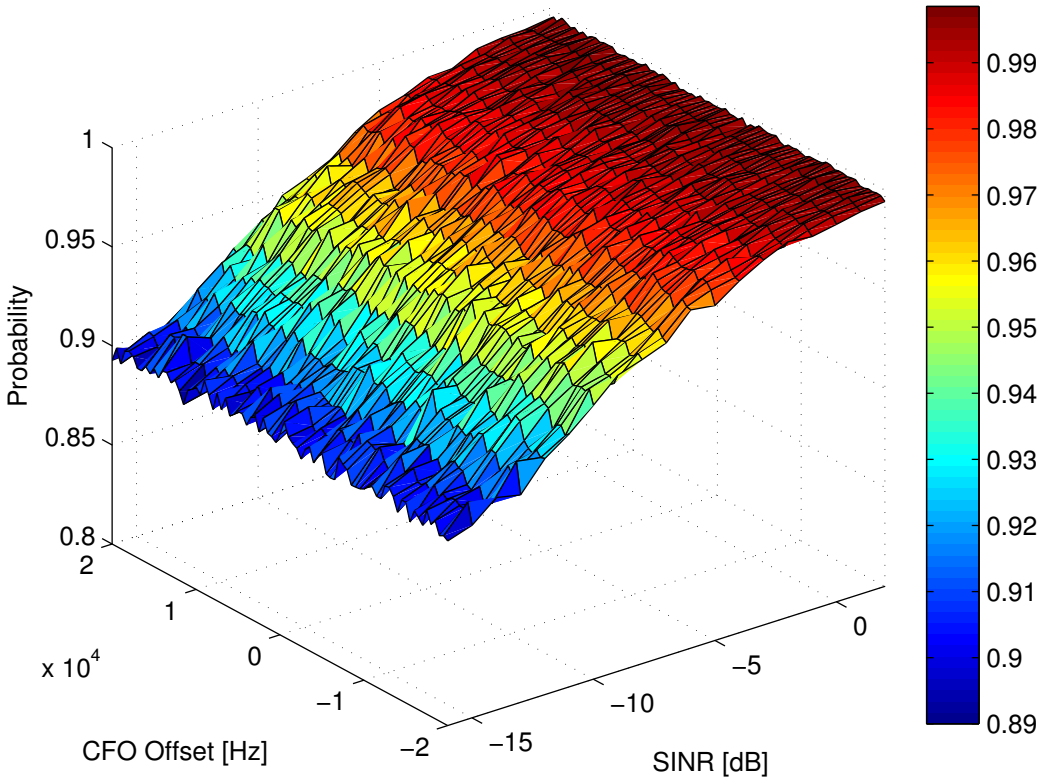
5.2.4 Detection Model Results

Results for the detection model are presented in Figures 5.2.2, 5.2.3, and 5.2.4. Figure 5.2.2 shows the probability of detecting that a PSSS is in a symbol given that there exists a PSSS in the symbol.

As we can see from Figure 5.2.2, given our target false alarm rate of 2.5%, the missed detection rate on detecting that a PSSS is present is fairly low even for a SINR as low as -16 dB. We can see that this missed detection rate does not depend on the carrier frequency offset. This tells us that our choice of PSSS hypotheses being spaced half a subcarrier or 7.5 kHz is sufficient.

Figure 5.2.3 shows the probability of detecting the correct PSSS given that a PSSS was detected. Again, we confirm that our choice of PSSS hypotheses being spaced half a subcarrier apart is sufficient due to the lack of variation in probability

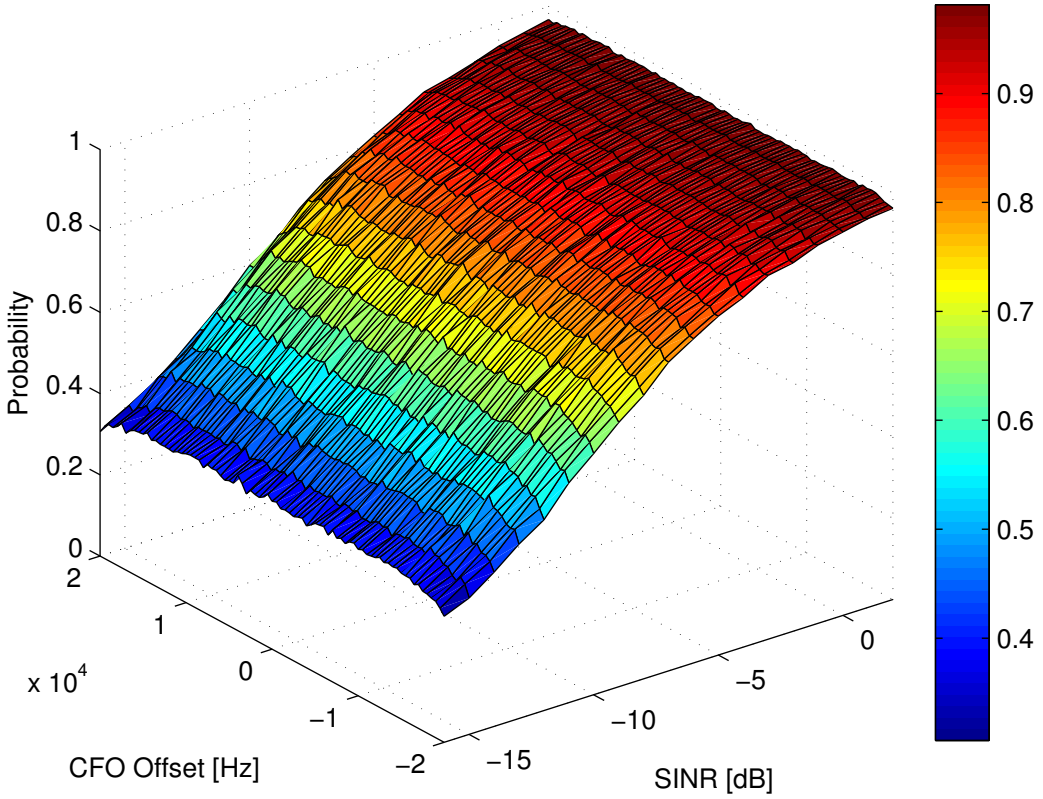
Figure 5.2.2: PSSS Presence Accuracy



with carrier frequency offset. We see that for a SINR of -16 dB we end up selecting the correct PSSS only 35.6% of the time. This is fairly poor performance considering that choosing at random would select the correct PSSS 25% of the time.

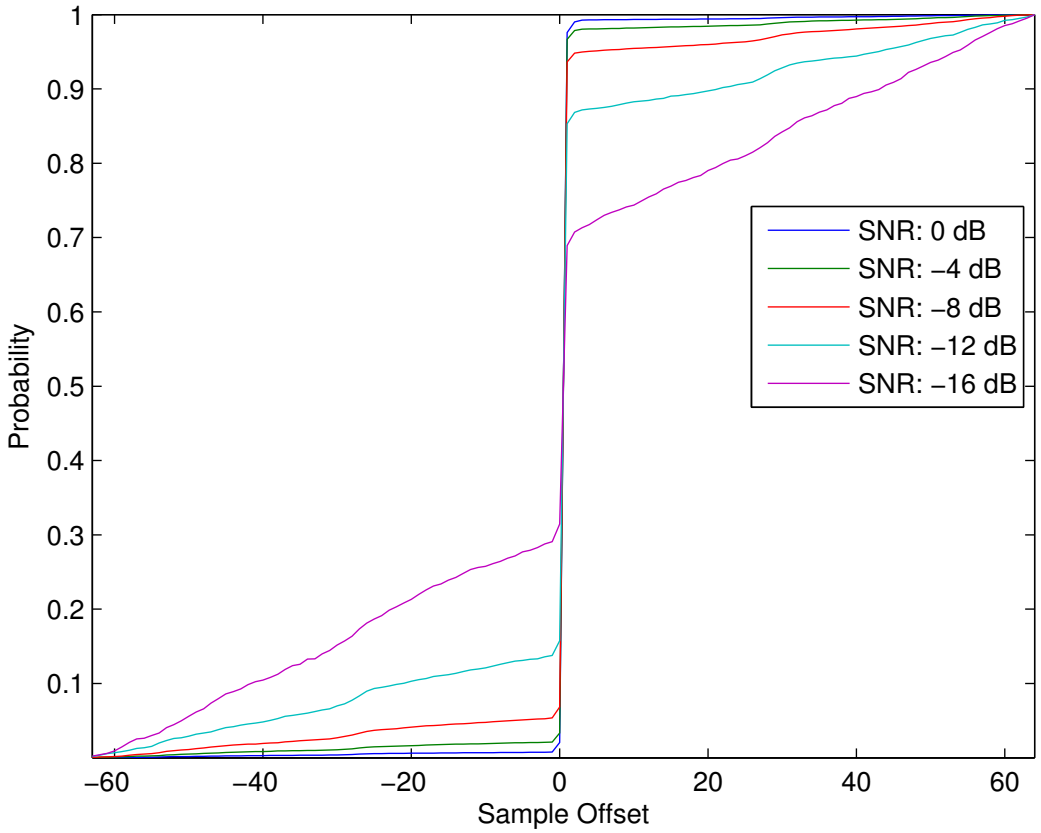
Figure 5.2.4 shows the probability cdf of how far the estimated sample offset is relative to the true sample offset. As expected, we see performance decline significantly at -16 dB which is the lower end of the SINR range we tested for. Based on the shape of the cdf, we can conclude that if the estimated sample offset is not accurate within a sample, then the difference between the estimate and the true value acts a uniform random variable across the set of possible offsets.

Figure 5.2.3: Correct PSSS Choice



The false alarm rate for detecting the presence of the PSSS can be adjusted to affect the performance of choosing the correct PSSS and the accuracy of the offset determination. However, without a good penalty metric, it is difficult to justify choosing any particular false alarm rate over another.

Figure 5.2.4: Sample Offset Determination



Chapter 6

Link Layer Analysis

In Chapter 5, we discuss a physical layer mechanism to detect a synchronization source based on PSSS detection. Specifically, we design for the ability to detect multiple PSSS transmission simultaneously within some SINR constraints. In this chapter, we will discuss some link layer design and simulations that utilize results from Chapter 5.

6.1 Design Considerations

Our proposed link layer utilizes the physical layer synchronization method proposed earlier in Chapter 5.

6.1.1 Desired Properties

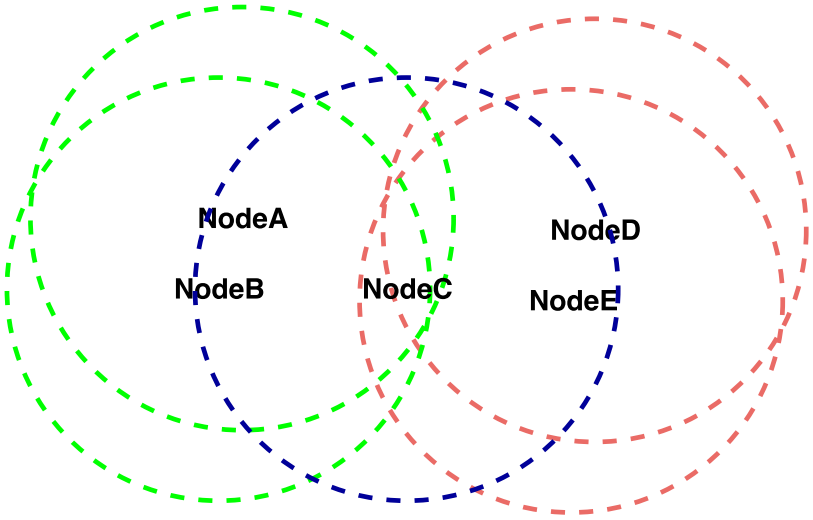
We introduce a few characteristics our link layer design should have which are important for any distributed synchronization system in order to function well.

The first goal is that the carrier frequency, sampling clock frequency, and sense of system timing for all nodes converges to a common value. It is not particularly important that the UEs converge on values that match true values for these quantities, but rather that they agree on them within some error margin.

The introduction of a new node in an unsynchronized state should not drastically impact synchronization states of other nodes. This includes nodes which are fully synchronized and those in almost synchronized states.

Acquisition and maintenance of synchronization state should be robust enough to allow a subset of the synchronization sources to drop out occasionally. There can be scenarios where there exist ”critical” nodes whose operation as a synchronization source is absolutely essentially to bring the system to a common synchronization state. In Figure 6.1.1, we see that Node C is ”critical” in the sense that without the presence of Node C, there is no physical mechanism that would allow synchronization information to pass between Nodes AB and Nodes DE. If Node C drops out of the network, it is only a matter of time before the synchronization states between the network formed by Nodes AB and the network formed between Nodes DE will diverge. However, the synchronization state nodes in the network should be able to tolerate Node B dropping out of the network formed by Nodes

Figure 6.1.1: Example Network



A,B,C,D, and E.

6.1.2 Effects of Propagation Delay

Timing offset estimation performance was previously discussed in Section 5.2.4. However, what was unmentioned is that there will exist propagation delay between a synchronization source and a receiver. For a 1 km distance at our base sampling rate, this equates to about 6.4 samples. If a receiver simply adjusts its sense of symbol start time using the offset it estimated from a synchronization source, it's sense of symbol start time will be ahead of the original synchronization source by the duration of the propagation delay assuming there are no other timing errors introduced in the system. In a simplified case where we have two nodes forming a network and taking turns as synchronization sources and receivers, this results in positive feedback where the symbol start time from a global perspective keeps

advancing by the propagation delay at every update.

In normal LTE operation, a UE acquires the downlink synchronization timing via the PSS/SSS transmissions. This allows the UE to decode a set of system parameters for the PRACH or Physical Random Access Channel. The PRACH is a physical channel where a UE sends a different type of synchronization signal to the eNB. The only purpose of this synchronization signal is to determine how many samples of propagation delay there are for UE-eNB transmissions. The eNB then sends a RAR or random access response that informs the UE how many samples it needs to advance its timing so transmission to the eNB arrived at some expected time [4].

While adopting something similar to the PRACH might be viable, we don't wish to adopt this method since it requires a response to every synchronization source in order to determine a timing advance. In addition, the synchronization source would then need to schedule downlink traffic for every timing advance request. PRACH processing is also one of the most computationally intensive operations for an eNB as it requires sample level search using cross correlation with a fairly large set of sequences over a long duration of time. PRACH also requires a contention resolution protocol as there is some probability for an unresolvable collision. All this simply adds a layer of complexity that would make a synchronization system very unwieldy.

6.1.3 Transmission Scheduling

Although we expanded the set of PSSS sequences in order to allow up to 4 simultaneous transmissions, we need to schedule synchronization transmissions in order have more than 4 synchronization sources. We want to schedule when nodes should act as synchronization sources to allow all nodes to participate in the network. In addition, scheduling needs to address the fact that synchronization sources can only operate in half duplex mode. They can not simultaneously act as a transmitter and receiver of synchronization signals. This is possible in a point to point system that uses FDD, but impossible since we are explicitly designing for a multipoint to multipoint system.

If synchronization sources have a transmission schedule that is known or predicted prior to synchronization, UEs in synchronization acquisition mode can potentially use combine estimates over a larger time frame. This is particularly important for getting synchronization information from farther away nodes in the presence of noise or interference from nearby nodes where there is only occasional successful decoding.

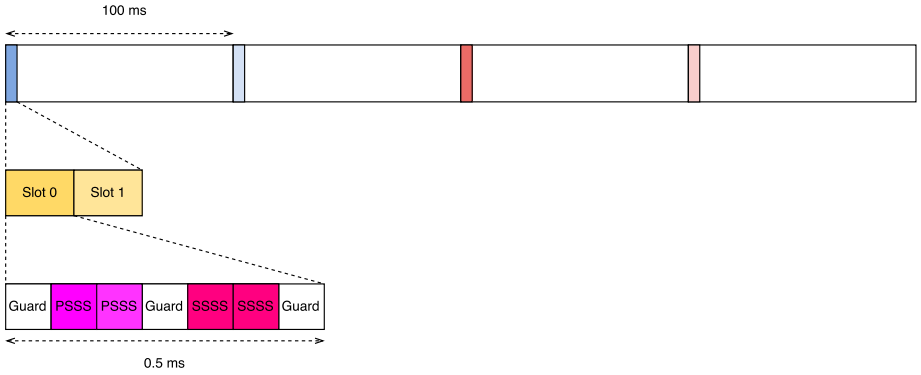
6.2 Proposed Design

6.2.1 Framing Structure

Our proposed design utilizes the framing structure as seen in Figure 6.2.1. A 1 ms subframe every 100 ms is designated as a synchronization subframe. Within each

synchronization subframe are two synchronization slots. Up to 4 nodes can participate in each synchronization slot.

Figure 6.2.1: Framing Structure for Distributed Synchronization



This design has the unique feature of being able to accommodate different numbers of nodes. With some higher layer help, our framing structure is flexible enough to keep utilization of the synchronization slots at a high rate. We propose that nodes be assigned positions within a synchronization subframe. A given node will utilize one particular PSSS sequence and consistently choose the same position in the framing structure for its PSSS transmissions. Hence, for one subframe up to 8 nodes can participate. However, depending on the system configuration, in the next frame, either up to another unique 8 nodes would participate or the same 8 nodes would once again participate. In theory this concept can be repeated over and over again, but there are practical limitations to how many nodes can realistically participate in the network. We propose nominally limiting the number of simultaneous synchronization sources to 32.

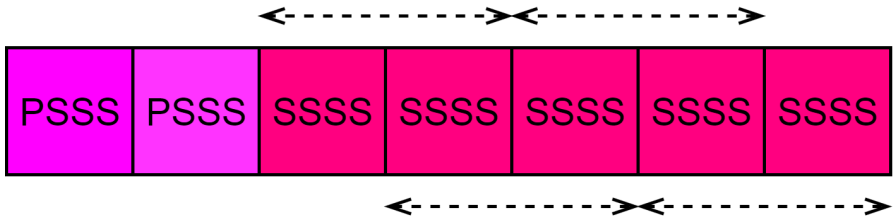
A UE processing this framing structure knows that a synchronization source is

guaranteed to periodically transmit once every 4 synchronization subframes. As such, without acquiring system information about the number of synchronization nodes the system is designed to support, a UE can combine synchronization metrics over longer periods of time to come up with a better estimate of the synchronization parameters. A UE can always assume that the system mode is the state for 32 nodes until it has acquired enough information to discern the actual number of synchronization nodes the system is configured for. In the initial mode from a node's point of view, a single user becomes possibly two or four. Until the number of users is successfully determined, there is a penalty for this flexible frame structure as it separates sets of synchronization metrics which could possibly be combined for form better estimates.

Guard intervals exist within this frame structure in order to handle transmission delay. Delayed transmission can cause power from the previous OFDM symbol of a distant node to bleed into the current symbol by a nearby node. Our framing structure currently makes a key assumption which is that the SSSS can also be decoded in a fashion similar to the PSSS. We envision a modified SSSS to operate similarly to the modified PSSS except using a different mathematical sequence as its basis for construction. We offer up a few alternative framing formats for the slot itself given the unknowns for what a modified SSSS would actually need.

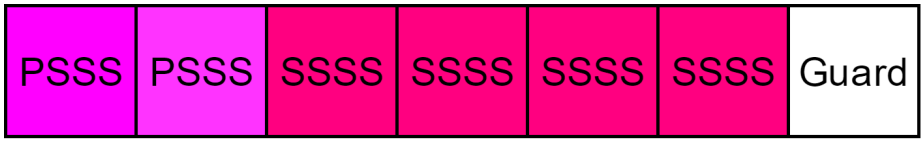
The alternative slot structure shown in Figure 6.2.2 removes all guard intervals. This assumes the acquisition of synchronization is robust enough to tolerate collisions due to different propagation delays plus some symbol start time synchronization errors. In this case, we still assume that the SSSS can take

Figure 6.2.2: Alternative Slot Structure 1 for Distributed Synchronization



advantage of a repeated symbol structure. Although there is overlap between what would be the modified SSSS waveform from different synchronization sources, it is minimized to some extent. In particular, the signals corresponding to the the first two SSSS symbols and the last two SSSS symbols do not have complete overlap with other SSSS signals. This can be prioritized for use to achieve better SINR, or some type of alternating assignment could be used to make average SINR for all synchronization sources after some number of synchronization subframes.

Figure 6.2.3: Alternative Slot Structure 2 for Distributed Synchronization



Another alternative slot structure is show in Figure 6.2.3. We can assign each SSSS symbol to a particular synchronization source or perhaps the SSSS could somehow utilize all 4 symbols and would overlap in time. One important thing to note is the presence of a guard symbol at the end of the slot. This guard symbol would protect the PSSS that would be located in Slot 1 as well as the first OFDM symbol of the subframe directly after the synchronization subframe. There would most likely need to be a guard symbol between the synchronization subframe and

other subframes. Ideally, this physical layer synchronization alleviates the requirements protection against interference in the other "normal" subframes. Whether that guard symbol is a part of the synchronization subframe or at the beginning of some "special" subframe which occurs directly before or directly after a synchronization subframe does not particularly matter.

Figure 6.2.4: Snapshots of ITU Ped A

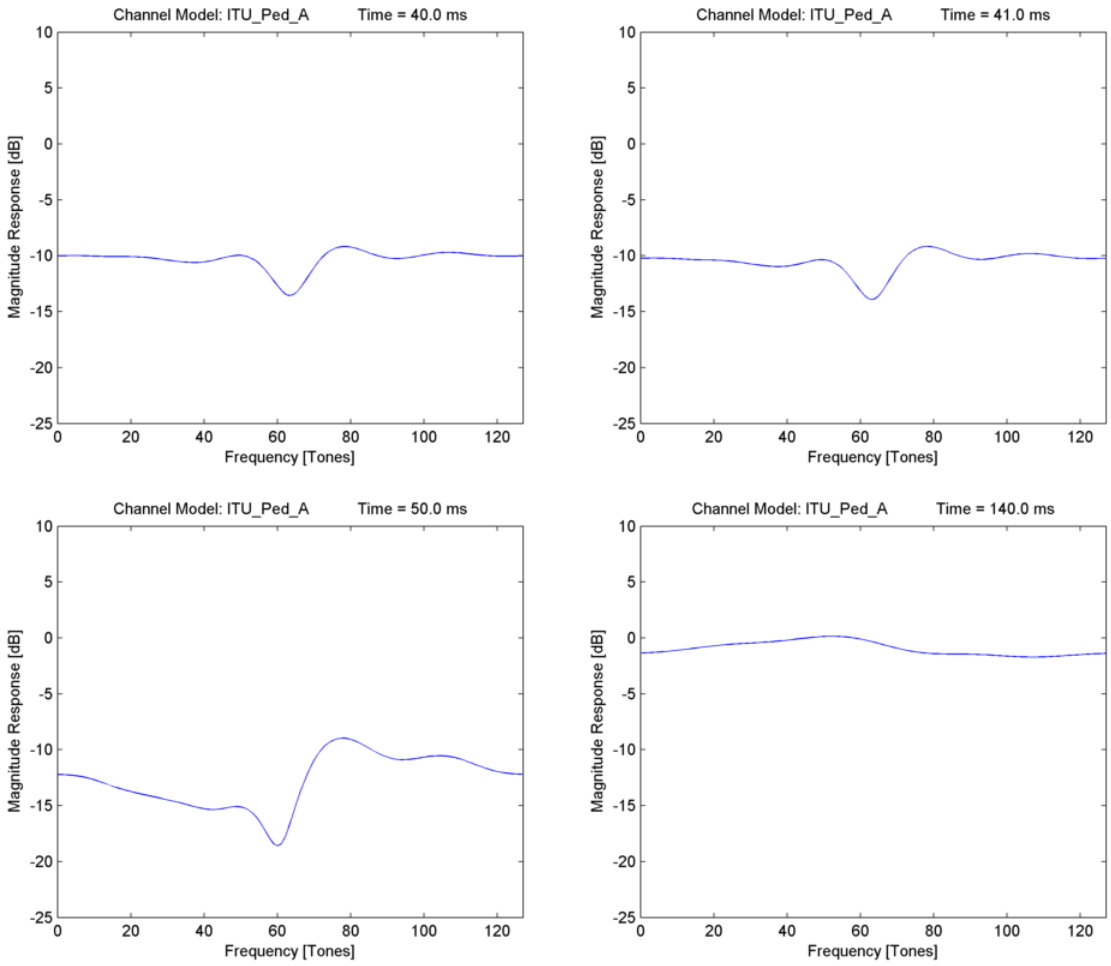


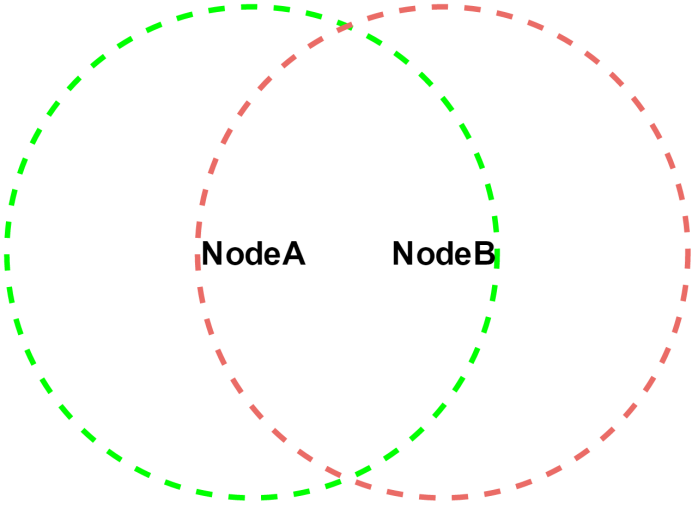
Figure 6.2.4 shows various snapshots of the magnitude response of an instance of the ITU Pedestrian A channel. The logic behind spacing synchronization subframes

100 ms apart (possibly more depending on the number of nodes the system mode is set for) is so if a particularly bad instance of the channel is seen at a particular point in time, there are enough other measurements taken at instances where the new channel is essentially uncorrelated with the bad instance that occurred previously. We see a instance of a bad channel at one particular point in time as it evolves over 1, 10 and 100 ms. Empirical observation confirms that this is fairly typical behavior of ITU Pedestrian A if the channel is in a bad state. If the expected channel we were designing the system for had a more aggressive Doppler shift, this could perhaps change the design decision to space the synchronization subframes 100 ms apart.

6.2.2 Timing Adjustment

As discussed previously in Section 6.1.2, simply using the estimated timing offset measured would result in positive feedback with respect to system timing as a whole. One approach to solving the generic problem of clock offsets is Gradient Time Synchronization Protocol (GTSP) [19]. The core of GTSP is that nodes can converge on a consensus for system time by averaging the timing offsets of received synchronization messages. GTSP does not deal with the effects of propagation delay and is designed for a mostly connected network. However given those constraints, all nodes will indeed converge to a consensus time whose error is bounded by the precision of the synchronization messages themselves. The idea of simply averaging the timing offsets is compelling since it is relatively easy to implement and converges in good operating scenarios.

Figure 6.2.5: 2 Node Scenario



We see in Figure 6.2.5, the simplest example scenario possible. Node A is already broadcasting synchronization signals periodically. Node B turns on and initializes with some sense of symbol start time at a random offset. In this case, Node B can apply the GTSP philosophy of ”averaging” the synchronization messages of Node A across Nodes. Although Node A is only one node, it could be operating in either 8 or 16 framing mode while Node B which is synchronized initializes by default assuming the 32 node framing format.

We can consider Figure 6.2.5, again with a slightly different scenario. In this scenario, two nodes A and B are already transmitting synchronization signals and come into communication range with each other. If Nodes A and B synchronously update their sense of system timing, there would be a ping pong effect. One simple modification in order to guarantee convergence is to modify simple averaging of estimated timing offsets to including a node's own timing offset (which would be zero from its perspective) into its updated computation of system time.

Averaging over long enough periods of times begins to break our assumption that within the synchronization period, sampling frequency offset is not significant enough to take into account. For the case of 20 ppm offset we are designing for, this results a 38 sample slip or jump over the period of 1 second at our base sample rate of 1.92. This is significant as it is almost a third of an OFDM symbol over a period of a second. We address this by proposing that nodes synchronously update at every baseline periodic repetition. That is every 400 ms will trigger a synchronous updates for nodes that are already synchronized or those that are in the process of synchronization. This limits the amount of sample slip or jump to 15 samples per synchronization update cycle.

We can further reduce sample slip/jumps from sampling frequency error by grouping synchronization subframes closer together. However, if we wish to keep the overall overhead the same, this requires leaving large stretches of time without any synchronization subframes. As discussed previously, keeping all synchronization subframes too close together could result in using a bad instance of a channel for a lot of transmissions.

In order to reduce the effects of propagation delay, we adopt a simple heuristic which can be adjusted depending on the typical network topology. Instead of simply averaging the set of estimates (including one's self) for the timing offset, we include an artificial weight, α_i, in our computation.

$$\Delta t = \frac{\sum_{i=0}^{N-1} \alpha_i \Delta \hat{t}_i}{\sum_{j \in S} \alpha_i} \tag{6.1}$$

where $\Delta\hat{f}_i$ is the timing offset estimate corresponding to the i$^{\text{th}}$ synchronization source, and N is the number of synchronization sources (including one's self). We set α to be 2 for the half of synchronization sources with more negative estimates and we set α to be 1 for the half of synchronization sources the rest of the synchronization estimates. Effectively, this weighs nearby nodes greater than farther away nodes. Effectively, this reduces the average propagation delay included in the overall timing esimate. α can be adjusted to be as aggressive possible by being equal to 1 for the closest node and 0 for all other nodes, i.e. relying on only the timing estimate of the closest node in order to minimize the effects of propagation delay to the minimum propagation delay of all synchronization sources.

6.2.3 Carrier Frequency Adjustment

We can apply a similar averaging algorithm in order to obtain an estimate for carrier frequency adjustment. Again, we take the philosophy that synchronization sources closer should be weighted more in the averaging calculation. The rationale behind this is that closer nodes will generally have better SINR and hence the probability that the validity and precision of the estimate will be higher. However, we still need to give weight to the farther away nodes in order to force the network to converge to common timing.

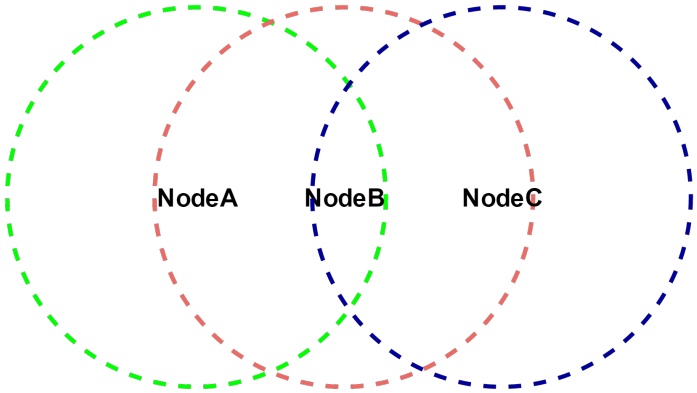
$$\Delta f = \frac{\sum_{i=0}^{N-1} \alpha_i \Delta\hat{f}_i}{\sum_{j \in S} \alpha_i} \tag{6.2}$$

Although we did not discuss an explicit algorithm for carrier frequency adjustment in Chapter 5, we suggest a framework for estimating carrier frequency offset. The determination of which PSSS sequence(s) is being used in a particular synchronization subframe will gives an initial estimate of carrier frequency offset to within half of a subcarrier. After determining the initial estimate of the offset to within half a subcarrier, the process can be repeated with PSSS hypotheses generated at smaller frequency intervals. The SSSS can also be used to provide information which can be incorporated into the frequency estimate. In addition, depending on the spacing between synchronization frames, the amount of sample slippage or jump due to the sampling clock and carrier clock being tied to the same local oscillator can also be used in conjunction with the other information in order to form some joint estimate of the local oscillator offset.

6.2.4 Choosing Synchronization Slot and PSSS Root

After a node decides to become a synchronization source, it must select a slot to transmit its synchronization signal and the PSSS waveform it is transmitting. We adopt a carrier sense multiple access mechanism (CSMA) . Nodes will initially listen for what synchronization sources are present. After acquiring sufficient timing to participate as a synchronization source, it will choose the slot being used by the least number of synchronization sources and an unused PSSS root within that slot. This mechanism is vulnerable to the hidden node problem which we demonstrate in Figure 6.2.6

Figure 6.2.6: Hidden Node Scenario



Under the CSMA mechanism, it is possible for Nodes A and C to choose the same synchronization slot and the same PSSS waveform since they are not in synchronization range of each other. Hence there can be a collision at Node B. We suggest two possible methods of preventing or remedying the situation.

The first is that nodes need undergo registration prior to becoming a synchronization source. Upon registration, all synchronized nodes within range will broadcast their neighbor list of synchronization sources which would include the slot and PSSS root number being used. This is somewhat analogous to PRACH where a UE transmits a signal for initial registration with the eNB which is followed by an acknowledgement along with an initial assignment of uplink resources in order to continue registration.

Another alternative is to do nothing at the physical layer and ultimately resolve the issue at a higher layer. If Node B is a synchronization source, eventually Nodes A and C will converge onto Node B's sense of timing. As long as Node B can detect the collision, it can broadcast higher layer messages informing Nodes A and C of a

collision.

6.2.5 Becoming a Synchronization Source

A node will decided it is in an adequate synchronization state if the its carrier frequency offset and symbol timing offset update magnitudes go below some predetermined threshold. After that, it will use the suggested CSMA mechanism in order to choose an appropriate slot and PSSS waveform to use.

6.3 Simulation and Results

In our scenarios, we generally assume that interference is the principal factor in SINR. This is reasonable if there are enough nearby nodes whose transmissions overlap. We assume the max SINR possible is 2 dB. This is fairly aggressive given the typical operating SNR for LTE devices. We see from Figures 5.2.2, 5.2.3, and 5.2.4 that the performance of our detection method is already close to the asymptote representing error free detection of PSSS and estimation of symbol start time offset at 2 dB so assuming a higher max SINR wouldn't change results much.

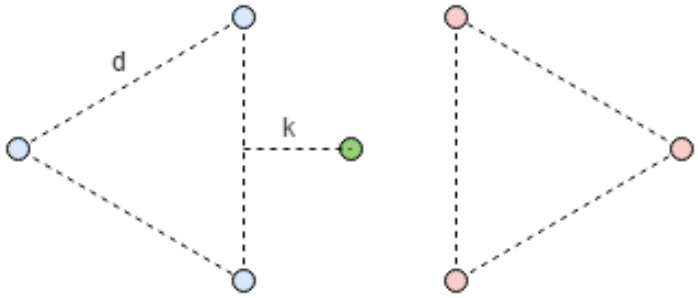
Distance in represented in terms of path loss between UEs. We use the free space path loss equation [18].

$$\text{Loss} = 20\log_{10}(d) + 20\log_{10}(f) + 20\log_{10}(\frac{4\pi}{c}) - G_t - G_r \tag{6.3}$$

where d is distance, f is the carrier frequency, c is the speed of light, and G is the antenna gain for either the transmitter or the receiver. We can lump all terms except for d into a generic constant since that loss is uniformly to all node to node pairs. The only unique loss is based on distance and since we have already determined that interference is the dominant factor, this constant applies equally to both interference and the relevant signal. We frame our distances in terms of a generic distance d.

6.3.1 Dual Triangle Scenario

Figure 6.3.1: Example Scenario



We have two disparate networks which have already agreed to their own system timings. The green node in the center comes online and we want to analyze how the system timing converges. The distance to the closer two nodes from the new node is $\sqrt{k^2 + \frac{d^2}{4}}$. The distance to the farther away node is $k + \frac{d}{2}\sqrt{3}$. We assume both networks are running in 16-node mode. In 16-node mode, there are 4 time slots and hence within each disparate network, there is no collision for the synchronization signal. We consider random permutations of slot selection among the two networks in order to generate interfering synchronization signals at the green node when the

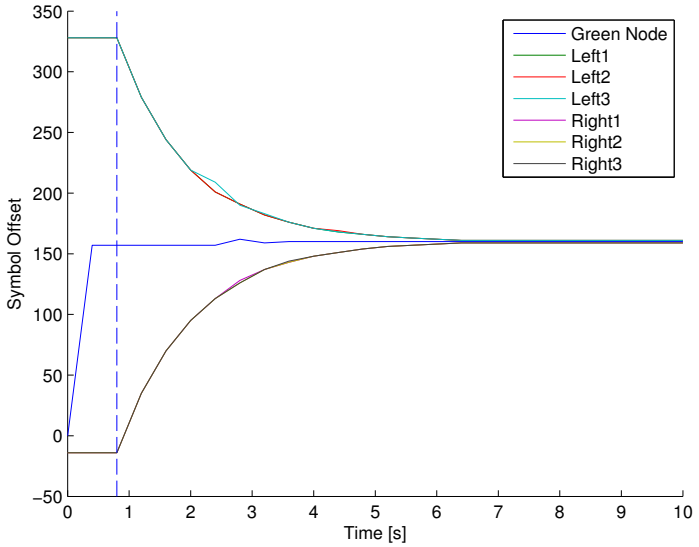
network converges.

In simulation we initialize with random symbol start time offset for the two disparate networks as well as random symbol offset. We take an example using $k = \frac{d}{2}$, but for some reason there is no viable RF path between the two disparate networks. In this scenario, the green node initially acquires a sense of symbol offset time and frame offset that is some average of the two networks. The farther away nodes are 6 dB down relative to the two closer nodes for each network. -6 dB still results in correct symbol detection and PSSS decodes approximately 81% of the time. For that successful 81% of decodes, the correct symbol start offset is estimated within a sample 93% of the time. These numbers are even higher for the closer nodes. We considered sample slippage and propagation delay to be minor enough to disregard.

Once the green node decides to become a synchronization source the synchronization parameters of the two networks begin to drift towards each other and the green node's parameters. Although the chance for decoding the closer nodes is extremely high, occasionally there is a miss which results in the average system synchronization parameters actually ending up biased towards one particular network's original parameters. The general speed of convergence is dependant on tunable parameters α, the threshold a node uses to determine its "synchronized enough", and the amount of inertia we wish to assign to already synchronized networks.

Figures 6.3.2 and 6.3.3 shows the convergence of the network's synchronization

Figure 6.3.2: Typical Symbol Offset Convergence of Scenario



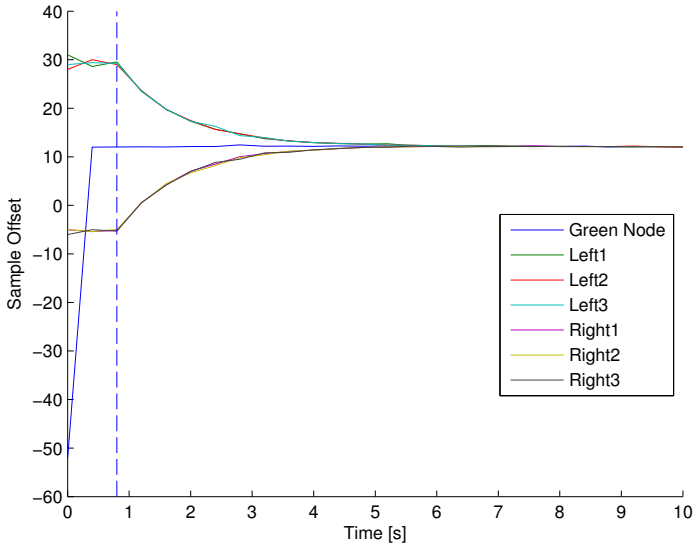
parameters for a typical instantiation of the scenario. The dotted blue line represents when the green node decides to become a synchronization source.

6.3.2 Random Scenario

We investigate various instances of ”random” initialization where nodes are placed randomly within some bounded box. We placed no restrictions on how close nodes could be to each other since we rely on arbitrary distance units. Again, we make this assumption since we believe that interference will be the principal factor in the denominator of SINR for these scenarios.

We show a typical example scenario in Figures 6.3.4 which is initialize with a random topology. The four nodes filled in represent a set of unsynchronized nodes which are already acting as synchronization sources. Again, we see good

Figure 6.3.3: Typical Sample Offset Convergence of Scenario



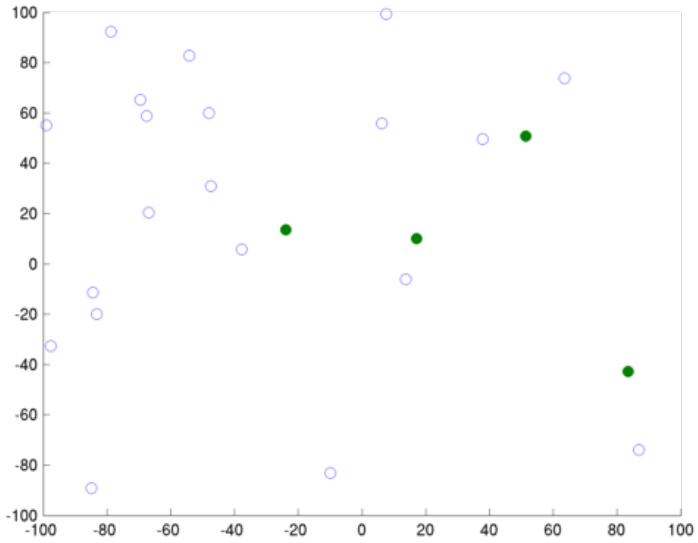
convergence behavior.

6.3.3 Observations

The scenario specified in Section 6.3.1 should be one of the more difficult scenarios for a synchronization protocol to resolve. Through several trials, some interesting observations were made which would impact further design of the protocol.

In this group of simulations, there would occasionally be collisions where the same PSSS waveform was transmitted from two different sources and collided at a receiver as the system timing converged. Due to the number of nodes in the system and the fact that the green node was in radio range of all the other nodes in the two networks, there was enough redundancy among the nodes to handle the issue

Figure 6.3.4: Typical Random Topology

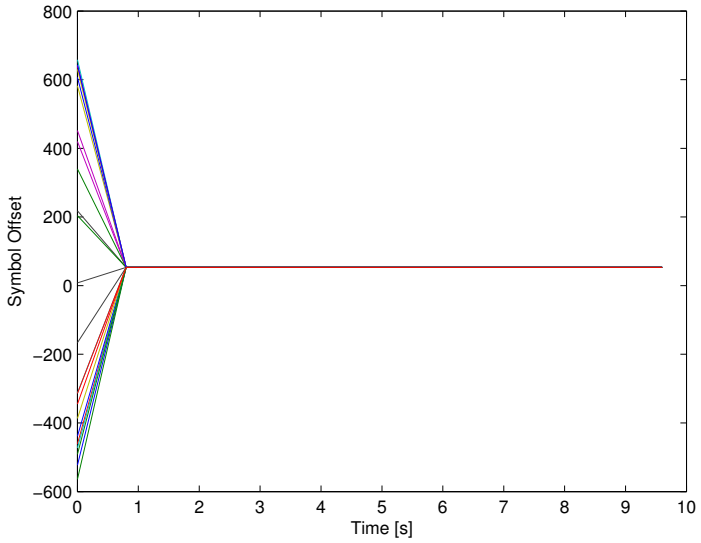


without any higher layer intervention or special physical layer protocol.

We see in the set of scenarios specified in Section 6.3.2, that for a typical random scenario we get good convergence. Without a bound on how close nodes can be, it is possible for nodes extremely close together to dominate a synchronization slot. However due to the fact there are up to 8 synchronization slots (depending on the mode), the network will still converge fairly quickly.

In the scenario shown in Figure 6.3.7, we hypothesize a full synchronized network which is in a line topology. Each node is spaced far enough away such that they can only listen to their two closest neighbors. If a new node to the left of the left most node is introduced, with static timing the timing of all the nodes in the line network will slowly drift towards this new node's timing. In particular, the leftmost node will generally be closest to the new node's parameters dragging its neighbor, etc. However, there is a time delay in propagating the new synchronization parameters

Figure 6.3.5: Typical Symbol Offset Convergence of Scenario



to the nodes farther right.

Due to the large gap in between our synchronization subframes, it could be considered problematic if the propagation of synchronization information results neighbors losing synchronization state. There is inherent inertia from our design due nodes counting themselves in the averaging of the estimates along with the fact that there are other nodes in the system transmitting synchronization information which is being taken into account through averaging. However, it is of interest to determine if updates for stable networks need to have some hard threshold for the amount of change each update can produce in order to maintain some minimum synchronization state with the nodes that is is already synchronized with. The exact bounds for defining whether two nodes are ”synchronized well enough” is difficult to determine without knowledge about what the rest of a full ad-hoc protocol would need for sufficient performance.

Figure 6.3.6: Typical Sample Offset Convergence of Scenario

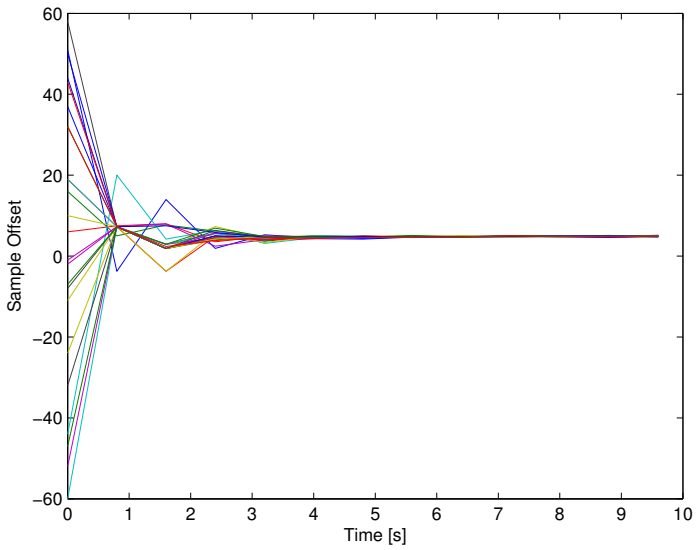
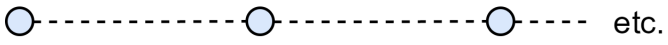


Figure 6.3.7: Line Scenario



Chapter 7

Conclusion and Final Thoughts

In this thesis, we presented an original implementation of a distributed physical layer synchronization protocol. This proposal reuses and expands upon existing physical layer signalling already present in LTE and is intended inform possibilities for retooling LTE synchronization for device to device communications. In particular, the novelty of our physical layer synchronization method is that we allow for simultaneous transmission and decoding of physical layer synchronization signals which occupies the same time-frequency resources. This potentially alleviates some of the scaling burdens associated with the current state of device to device LTE synchronization which relies exclusively on assigning time slots to one particular user.

Using performance curves generated for our physical layer synchronization method, we abstract the performance curves to probabilistic events for study at the link layer. We consider several design parameters and design choices that would

need to be made at the link layer which are generic to distributed synchronization as well as specific problems which pertain only to our physical layer synchronization method. We implement some of our link layer design in simulation in order to flesh out these design choices as well as make general observations about system behavior. Hopefully, some of the design aspects and core ideas for either the physical layer design or link layer design can provide some insight in the ongoing research efforts into ad-hoc networking.

7.1 Future Work

The core principle of the physical layer design is that multiple users can use the same time-frequency resources and successfully acquire symbol level timing, a rough sense of carrier frequency offset, as well sample level timing. Although the physical layer design is fairly fleshed out, the link layer design actually leaves a lot of freedom for design choices. There are many parameters where we seem to have to make trade offs in terms of achieving the system synchronization goals and it is unclear what a system designer should actually prioritize. The physical layer design currently leaves room for a secondary synchronization signal which would either be able to utilize the initial information discerned from the primary or perhaps could be decoded in some joint fashion. From a link layer point of view, there are too many design choices for physical layer parameters to do an exhaustive search for the best parameters. Without some type of quality function or function to measure a success metric, it is hard to absolutely declare one set of parameters superior to the other.

The original code is implemented mostly in MATLAB, although there were some initial attempts for implementation in C which were quickly abandoned mostly for speed of development. Coming from a DSP background, I would want to implement some of the physical layer processing in optimized assembly on a DSP since our proposal places a very significant computational burden for the physical layer processing.

Appendix A

PSSS ML Detector Tables

As described in Section 5.1, the original ML estimator for PSS detection also applies to the PSSS as described in LTE Release 12. The cross correlation properties for the set of roots indices coprime with the length (63) is shown in Table A.0.1.

Table A.0.1: ML Detection Peaks for PSSS

Root	**2**	**4**	**5**	**8**	**10**	**11**	**13**	**16**	**17**	**19**	**20**
2	1.00	0.17	0.22	0.22	0.24	0.38	0.23	0.33	0.23	0.21	0.38
4	0.17	1.00	0.22	0.19	0.22	0.33	0.38	0.22	0.21	0.23	0.18
5	0.22	0.22	1.00	0.25	0.21	0.22	0.19	0.20	0.22	0.33	0.22
8	0.22	0.19	0.25	1.00	0.22	0.22	0.18	0.22	0.38	0.25	0.23
10	0.24	0.22	0.21	0.22	1.00	0.19	0.22	0.22	0.33	0.38	0.27
11	0.38	0.33	0.22	0.22	0.19	1.00	0.21	0.22	0.22	0.21	0.38
13	0.23	0.38	0.19	0.18	0.22	0.21	1.00	0.24	0.17	0.22	0.33
16	0.33	0.22	0.20	0.22	0.22	0.22	0.24	1.00	0.21	0.22	0.26
17	0.23	0.21	0.22	0.38	0.33	0.22	0.17	0.21	1.00	0.17	0.23
19	0.21	0.23	0.33	0.25	0.38	0.21	0.22	0.22	0.17	1.00	0.18
20	0.38	0.18	0.22	0.23	0.27	0.38	0.33	0.26	0.23	0.18	1.00
22	0.22	0.38	0.23	0.33	0.23	0.27	0.38	0.22	0.25	0.22	0.20
23	0.58	0.20	0.38	0.22	0.17	0.24	0.20	0.33	0.22	0.20	0.22
25	0.20	0.58	0.21	0.18	0.22	0.33	0.22	0.38	0.18	0.22	0.25
26	0.22	0.18	0.58	0.38	0.24	0.24	0.29	0.24	0.38	0.33	0.22
29	0.38	0.22	0.23	0.58	0.17	0.38	0.21	0.24	0.22	0.18	0.38
31	0.23	0.38	0.23	0.23	0.58	0.21	0.38	0.24	0.33	0.24	0.20
32	0.22	0.33	0.38	0.23	0.25	0.58	0.18	0.25	0.22	0.21	0.25
34	0.20	0.22	0.21	0.21	0.23	0.21	0.58	0.38	0.22	0.22	0.33
37	0.33	0.23	0.19	0.21	0.38	0.21	0.22	0.58	0.22	0.38	0.20
38	0.38	0.23	0.23	0.22	0.33	0.38	0.19	0.19	0.58	0.24	0.38
40	0.20	0.38	0.33	0.22	0.23	0.18	0.38	0.22	0.20	0.58	0.24
41	0.22	0.24	0.38	0.24	0.17	0.22	0.33	0.20	0.22	0.18	0.58
43	0.20	0.22	0.16	0.33	0.23	0.19	0.25	0.38	0.21	0.22	0.27
44	0.58	0.19	0.22	0.38	0.23	0.22	0.20	0.33	0.38	0.23	0.22
46	0.20	0.58	0.18	0.19	0.38	0.33	0.22	0.22	0.26	0.38	0.21
47	0.38	0.19	0.58	0.22	0.18	0.38	0.20	0.20	0.22	0.33	0.38
50	0.23	0.21	0.38	0.58	0.20	0.23	0.28	0.20	0.22	0.20	0.25
52	0.17	0.22	0.23	0.21	0.58	0.34	0.23	0.38	0.33	0.22	0.19
53	0.23	0.33	0.24	0.38	0.27	0.58	0.20	0.18	0.38	0.23	0.23
55	0.22	0.22	0.27	0.20	0.38	0.21	0.58	0.22	0.19	0.38	0.33
58	0.33	0.38	0.28	0.27	0.24	0.23	0.38	0.58	0.18	0.22	0.16
59	0.23	0.21	0.38	0.22	0.33	0.22	0.21	0.19	0.58	0.19	0.22
61	0.23	0.23	0.33	0.22	0.23	0.17	0.23	0.38	0.20	0.58	0.20

Root	**22**	**23**	**25**	**26**	**29**	**31**	**32**	**34**	**37**	**38**	**40**
2	0.22	0.58	0.20	0.22	0.38	0.23	0.22	0.20	0.33	0.38	0.20
4	0.38	0.20	0.58	0.18	0.22	0.38	0.33	0.22	0.23	0.23	0.38
5	0.23	0.38	0.21	0.58	0.23	0.23	0.38	0.21	0.19	0.23	0.33
8	0.33	0.22	0.18	0.38	0.58	0.23	0.23	0.21	0.21	0.22	0.22
10	0.23	0.17	0.22	0.24	0.17	0.58	0.25	0.23	0.38	0.33	0.23
11	0.27	0.24	0.33	0.24	0.38	0.21	0.58	0.21	0.21	0.38	0.18
13	0.38	0.20	0.22	0.29	0.21	0.38	0.18	0.58	0.22	0.19	0.38
16	0.22	0.33	0.38	0.24	0.24	0.24	0.25	0.38	0.58	0.19	0.22
17	0.25	0.22	0.18	0.38	0.22	0.33	0.22	0.22	0.22	0.58	0.20
19	0.22	0.20	0.22	0.33	0.18	0.24	0.21	0.22	0.38	0.24	0.58
20	0.20	0.22	0.25	0.22	0.38	0.20	0.25	0.33	0.20	0.38	0.24
22	1.00	0.18	0.24	0.20	0.33	0.38	0.21	0.22	0.22	0.19	0.38
23	0.18	1.00	0.19	0.23	0.24	0.25	0.38	0.19	0.33	0.23	0.23
25	0.24	0.19	1.00	0.20	0.21	0.22	0.33	0.38	0.23	0.18	0.23
26	0.20	0.23	0.20	1.00	0.26	0.23	0.22	0.21	0.24	0.23	0.33
29	0.33	0.24	0.21	0.26	1.00	0.20	0.24	0.22	0.21	0.38	0.19
31	0.38	0.25	0.22	0.23	0.20	1.00	0.17	0.24	0.22	0.33	0.38
32	0.21	0.38	0.33	0.22	0.24	0.17	1.00	0.20	0.23	0.22	0.25
34	0.22	0.19	0.38	0.21	0.22	0.24	0.20	1.00	0.26	0.21	0.24
37	0.22	0.33	0.23	0.24	0.21	0.22	0.23	0.26	1.00	0.20	0.23
38	0.19	0.23	0.18	0.23	0.38	0.33	0.22	0.21	0.20	1.00	0.19
40	0.38	0.23	0.23	0.33	0.19	0.38	0.25	0.24	0.23	0.19	1.00
41	0.27	0.38	0.19	0.22	0.22	0.21	0.38	0.33	0.20	0.24	0.18
43	0.58	0.24	0.38	0.20	0.33	0.25	0.20	0.38	0.22	0.25	0.22
44	0.18	0.58	0.24	0.38	0.22	0.21	0.24	0.18	0.33	0.22	0.20
46	0.22	0.20	0.58	0.22	0.22	0.22	0.33	0.22	0.38	0.18	0.22
47	0.20	0.22	0.19	0.58	0.38	0.25	0.24	0.25	0.24	0.38	0.33
50	0.33	0.38	0.19	0.22	0.58	0.18	0.38	0.21	0.29	0.22	0.20
52	0.23	0.18	0.38	0.21	0.22	0.58	0.21	0.38	0.24	0.33	0.24
53	0.17	0.23	0.33	0.38	0.23	0.25	0.58	0.17	0.24	0.22	0.17
55	0.24	0.22	0.22	0.21	0.21	0.23	0.23	0.58	0.38	0.18	0.22
58	0.38	0.33	0.23	0.19	0.21	0.38	0.23	0.23	0.58	0.21	0.38
59	0.24	0.38	0.23	0.23	0.22	0.33	0.38	0.22	0.18	0.58	0.20
61	0.22	0.20	0.38	0.33	0.20	0.22	0.23	0.38	0.22	0.20	0.58

Root	**41**	**43**	**44**	**46**	**47**	**50**	**52**	**53**	**55**	**58**	**59**	**61**
2	0.22	0.20	0.58	0.20	0.38	0.23	0.17	0.23	0.22	0.33	0.23	0.23
4	0.24	0.22	0.19	0.58	0.19	0.21	0.22	0.33	0.22	0.38	0.21	0.23
5	0.38	0.16	0.22	0.18	0.58	0.38	0.23	0.24	0.27	0.28	0.38	0.33
8	0.24	0.33	0.38	0.19	0.22	0.58	0.21	0.38	0.20	0.27	0.22	0.22
10	0.17	0.23	0.23	0.38	0.18	0.20	0.58	0.27	0.38	0.24	0.33	0.23
11	0.22	0.19	0.22	0.33	0.38	0.23	0.34	0.58	0.21	0.23	0.22	0.17
13	0.33	0.25	0.20	0.22	0.20	0.28	0.23	0.20	0.58	0.38	0.21	0.23
16	0.20	0.38	0.33	0.22	0.20	0.20	0.38	0.18	0.22	0.58	0.19	0.38
17	0.22	0.21	0.38	0.26	0.22	0.22	0.33	0.38	0.19	0.18	0.58	0.20
19	0.18	0.22	0.23	0.38	0.33	0.20	0.22	0.23	0.38	0.22	0.19	0.58
20	0.58	0.27	0.22	0.21	0.38	0.25	0.19	0.23	0.33	0.16	0.22	0.20
22	0.27	0.58	0.18	0.22	0.20	0.33	0.23	0.17	0.24	0.38	0.24	0.22
23	0.38	0.24	0.58	0.20	0.22	0.38	0.18	0.23	0.22	0.33	0.38	0.20
25	0.19	0.38	0.24	0.58	0.19	0.19	0.38	0.33	0.22	0.23	0.23	0.38
26	0.22	0.20	0.38	0.22	0.58	0.22	0.21	0.38	0.21	0.19	0.23	0.33
29	0.22	0.33	0.22	0.22	0.38	0.58	0.22	0.23	0.21	0.21	0.22	0.20
31	0.21	0.25	0.21	0.22	0.25	0.18	0.58	0.25	0.23	0.38	0.33	0.22
32	0.38	0.20	0.24	0.33	0.24	0.38	0.21	0.58	0.23	0.23	0.38	0.23
34	0.33	0.38	0.18	0.22	0.25	0.21	0.38	0.17	0.58	0.23	0.22	0.38
37	0.20	0.22	0.33	0.38	0.24	0.29	0.24	0.24	0.38	0.58	0.18	0.22
38	0.24	0.25	0.22	0.18	0.38	0.22	0.33	0.22	0.18	0.21	0.58	0.20
40	0.18	0.22	0.20	0.22	0.33	0.20	0.24	0.17	0.22	0.38	0.20	0.58
41	1.00	0.20	0.22	0.25	0.22	0.38	0.27	0.23	0.33	0.23	0.38	0.22
43	0.20	1.00	0.18	0.23	0.26	0.33	0.38	0.27	0.23	0.22	0.18	0.38
44	0.22	0.18	1.00	0.17	0.22	0.22	0.21	0.38	0.25	0.33	0.23	0.21
46	0.25	0.23	0.17	1.00	0.21	0.17	0.22	0.33	0.38	0.22	0.21	0.23
47	0.22	0.26	0.22	0.21	1.00	0.24	0.22	0.22	0.23	0.20	0.22	0.33
50	0.38	0.33	0.22	0.17	0.24	1.00	0.21	0.22	0.18	0.19	0.38	0.23
52	0.27	0.38	0.21	0.22	0.22	0.21	1.00	0.19	0.22	0.22	0.33	0.38
53	0.23	0.27	0.38	0.33	0.22	0.22	0.19	1.00	0.22	0.21	0.22	0.25
55	0.33	0.23	0.25	0.38	0.23	0.18	0.22	0.22	1.00	0.25	0.19	0.22
58	0.23	0.22	0.33	0.22	0.20	0.19	0.22	0.21	0.25	1.00	0.22	0.22
59	0.38	0.18	0.23	0.21	0.22	0.38	0.33	0.22	0.19	0.22	1.00	0.17
61	0.22	0.38	0.21	0.23	0.33	0.23	0.38	0.25	0.22	0.22	0.17	1.00

Appendix B

Relevant Source Code

B.1 Helper Functions

```
function [ out ] = zc_seq( R,N )
%Arguments:
%   R : scalar [root of ZC sequence]
%   N : scalar [of length N]
%   out: Nx1 [output sequence]

ind = (0:N-1).';
out = exp(-1i*pi*R*ind.*(ind+1)/N);

end


function [ out, delay ] = cxcorr( A, B )
%Arguments:
%   A : NxM vector
%   B : NxM vector

if ~(isequal(size(A), size(B)) || (isvector(A) && isvector(B) && numel(A) == numel(B)))
    assert(false,'Check sizes of inputs');
```

```
end

out = circshift(ifft(fft(A,[],1).*conj(fft(B,[],1))),floor(size(B,1)/2),1);
delay = -floor(size(B,1)/2):floor(size(B,1)/2);
end


function [ out ] = scfdma_mdm(A, mapping, nfft, cp_len,ismod)
%SCFDMA mod/demod for single user
%Arguments:
%   A : M x N vector (nfft x N vector) [scfdma mod input or demod input]
%   mapping: M x 1 vector (M x 1 vector) [map to freq grid]
%   nfft: scalar
%   cp_len: scalar [length of cyclic prefix]
%   ismod: true (false) [mod or demod]
%Output:
%  out: nfft x N vector (M x N vector) [scfdma mod output or demod output]

if (ismod)
    fgrid = zeros(nfft,size(A,2));
    fgrid(mapping,:) = fft(A,[],1);
    fgrid = fgrid ./ sqrt(numel(mapping));
    out = ifft(fgrid);
    out = sqrt(nfft)*out;
    %CP attachment
    ncp_left = cp_len;
    while(ncp_left > 0)
        if (ncp_left > nfft)
            out = [out;out];
            ncp_left = ncp_left - nfft;
        else
            out = [out(end-ncp_left+1:end,:);out];
            ncp_left = ncp_left - ncp_left;
        end
    end
else
    fgrid = A(end-nfft-1:end,:);
    fgrid = fft(fgrid,[],1);
    fgrid(mapping,:) = fgrid;
    out = ifft(fgrid,[],1);
```

```
end

end


function [ r ] = psss_detector(s, c)
%s : received time domain signal (Nx1)
%c : candidates (NxM)
%r : results. 1st col = index of c chosen. 2nd col = peak2avg.
ew_divide = @(a,b) a./b;

s_t = repmat(s,[1 , size(c,2)]);
m = abs(cxcorr(s_t,c));

[mx_val, mx_idx] = max(m);
%grab peak and zero out immediate neighbors for avg calculation
s_used = -2:2;
c_idx = mod(bsxfun(@plus,mx_idx,s_used.'),size(s,1));
c_idx(c_idx==0) = size(s,1);
c_idx = sub2ind(size(c),c_idx, repmat(1:size(c,2),[numel(s_used),1]));

m(c_idx) = 0;

mx_val2 = max(m);

m_mean = mean(m,1)*(size(m,1)/(size(m,1)-size(c_idx,1)));

% [metric,metric_idx] = max(mx_val./(m_mean.*mx_val2));
% [metric,metric_idx] = max(mx_val./(m_mean.*sqrt(mx_val2)));
[metric,metric_idx] = max(mx_val);
mx_val_offset = c_idx(ceil(numel(s_used)/2),metric_idx) - (metric_idx-1)*size(s,1);
r(1) = metric;
r(2) = metric_idx;
r(3) = mx_val_offset;
r(4) = mx_val(metric_idx);
r(5) = m_mean(metric_idx);
r(6) = mx_val2(metric_idx);
end
```

```
function [ raychan ] = ray_chan_generate(m,samp_rate,opt,report)
%Arguments:
%   m: # of channels to generate
%   opt: option to choose channel model
%Description: initializes a channel.rayleigh objects according to ITU channel model

%Opt;
% 'PedA' - PathDelay=[0, 110e-9, 190e-9, 410e-9]; Fd = 3;
% 'VecA' - PathDelay=[0, 310e-9, 710e-9, 1090e-9, 1730e-9, 2510e-9]; Fd =
%           30;


switch opt
    case 'PedA'
        PathDelay=[0, 110e-9, 190e-9, 410e-9];
        Fd = 3;
        AvgGaindB=[0, -9.7, -19.2, -22.8];
    case 'VecA'
        PathDelay=[0, 310e-9, 710e-9, 1090e-9, 1730e-9, 2510e-9];
        Fd = 30;
        AvgGaindB=[0, -1, -9, -10, -15, -20];
    otherwise
        display('Unknown channel type');
        return
end

raychan = cell(m,1);

for ii = 1:m
    raychan{ii} = rayleighchan(1/samp_rate,Fd,PathDelay,AvgGaindB);
    raychan{ii}.NormalizePathGains = 1;
    raychan{ii}.ResetBeforeFiltering = 1;
    raychan{ii}.StoreHistory = 0;
%     filter(raychan{ii},0);
    raychan{ii}.ResetBeforeFilter = 1;
end
```

B.2 PSSS Set Investigation

```
%Some functions
ew_mult = @(a,b) a.*b;
ew_divide = @(a,b) a./b;

%Parameters for 1.4MHz BW
base_sr = 1.92e6;
nfft = 128;
cp_len = 0;
ofdm_map = [29:100];

%Generate all valid ZC seqs
zc_len = 63;
zc_idx = 2:zc_len-2;
zc_idx = zc_idx(gcd(zc_idx,zc_len)==1);

A = zeros(zc_len,numel(zc_idx));
for ii = 1:numel(zc_idx)
    A(:,ii) = zc_seq(zc_idx(ii),63);
end
A(32,:) = [];
%Modulate
A_map = [zeros(5,size(A,2));A;zeros(5,size(A,2))];
s = scfdma_mdm(A_map, ofdm_map,128, cp_len, 1);

%Apply SFO
f_offset = 0;
t = (0:size(s,1)-1).'/base_sr;
s_fa = bsxfun(ew_mult,s, exp(1i*2*pi*f_offset*t));

%Apply ML detector for all pairs
m_metric = zeros(2*size(s_fa,1)-1,size(s_fa,2),size(s_fa,2));
for ii = 1:size(s_fa,2)
    for jj = 1:size(s_fa,2)
        m_metric(:,ii,jj) = xcorr(s_fa(:,ii),s(:,jj));
    end
end
```

```
q = squeeze(max(abs(m_metric),[],1));
q = bsxfun(ew_divide,q,diag(q));

%Apply Alternative detector for all pairs
m_metric2 = zeros(size(s_fa,1),size(s_fa,2),size(s_fa,2));
for ii = 1:size(s_fa,2)
    for jj = 1:size(s_fa,2)
        m_metric(:,ii,jj) = cxcorr(s_fa(:,ii),s(:,jj));
    end
end

%% Compute CM
cm = zeros(size(s,2),1);
for ii = 1:size(s,2)
   cm(ii) = (mag2db(rms((abs(s(:,ii))/rms(abs(s(:,ii)))).^3)) - 2.94)/2.45; %factors ba
end
```

B.3 Probability Simulation

```
addpath(genpath('./helpers'));

%Some functions
ew_mult = @(a,b) a.*b;
ew_divide = @(a,b) a./b;

%Parameters for 1.4MHz BW
base_sr = 1.92e6;
nfft = 128;
cp_len = 0;
cp_len_filt = 9;
ofdm_map = [29:100];

%Generate all valid ZC seqs
zc_len = 63;
zc_idx = 2:zc_len-2;
zc_idx = zc_idx(gcd(zc_idx,zc_len)==1);
```

```
A = zeros(zc_len,numel(zc_idx));
for ii = 1:numel(zc_idx)
    A(:,ii) = zc_seq(zc_idx(ii),63);
end
A(32,:) = [];

A = A(:,[find(zc_idx==4),find(zc_idx==26),find(zc_idx==37),find(zc_idx==59)]);

%Modulate
A_map = [zeros(5,size(A,2));A;zeros(5,size(A,2))];
s = scfdma_mdm(A_map, ofdm_map,128, cp_len, 1);

%Apply SFO for references
t = (0:size(s,1)-1).'/base_sr;
CFO_hypo = [-15e3:7.5e3:15e3];
s_hypo = zeros(size(s,1),numel(CFO_hypo)*size(s,2));

for ii = 1:numel(CFO_hypo)
    s_hypo(:,(ii-1)*size(s,2)+1:ii*size(s,2)) = ...
        bsxfun(ew_mult,s,exp(1i*2*pi*CFO_hypo(ii)*t));
end

%Set Params & Generate Channels
cfo_set = -20000:500:20000;
sinr = 2:-1:-16;

n_iter = 10000;
tic
rch = ray_chan_generate(n_iter,base_sr,'PedA');
time_a = toc;
d_results = zeros(numel(cfo_set),numel(sinr),size(s,2),numel(rch),6);
perc = 0;
h = waitbar(0,'Initializing waitbar...');
n_cfo_set = numel(cfo_set);
ng = zeros(size(s_hypo,1),numel(rch),numel(sinr));

%% Main Probability Sim

for jj = 1:numel(sinr)
   ng(:,:,jj) = noise_gen(size(s_hypo,1),numel(rch),zc_len,-sinr(jj));
```

```
end
for ii = 1:numel(cfo_set)
    s_f = bsxfun(ew_mult,s,exp(1i*2*pi*cfo_set(ii)*t));
    s_f = [s_f(end-cp_len_filt+1:end,:);s_f];

    for kk = 1:size(s_f,2)
        y = s_f(:,kk);
        for ll = 1:numel(rch)
            yt = filter(rch{ll},y);
            yt = yt(cp_len_filt+1:end);
            for jj = 1:numel(sinr)
                ytn = yt + ng(:,ll,jj);
                d_results(ii,jj,kk,ll,:) = psss_detector(ytn,s_hypo);
            end
        end
    end
        perc = ii/numel(cfo_set);
     waitbar(perc,h,sprintf('%d of %d / Init Time: %0.2f/ Curr Time: %0.2f',ii, n_cfo_se
end
%% Probabilities for PSSS and Symbol Correctness
d_max = zeros(size(d_results,1),size(d_results,2),size(d_results,3),size(d_results,4)
d_max(:,:,:,:) = squeeze(d_results(:,:,:,:,4));

d_mean = zeros(size(d_results,1),size(d_results,2),size(d_results,3),size(d_results,4
d_mean(:,:,:,:) = squeeze(d_results(:,:,:,:,5));

d_rat = d_max./d_mean;

% d_pstrat = (d_rat > 2.859); %95 percentile
d_pstrat = (d_rat > 2.95); %97.5 percentile
% d_pstrat = (d_rat > 3.06); %99 percentile


d_idx = zeros(size(d_results,1),size(d_results,2),size(d_results,3),size(d_results,4)
d_idx(:,:,:,:) = squeeze(d_results(:,:,:,:,2));
d_idx = mod(d_idx,size(s,2));

d_idx(d_idx==0) = size(s,2);

q = zeros(1,1,size(s,2),1);
```

```
q(1,1,:,1) = 1:size(s,2);
q = repmat(q,[size(d_idx,1),size(d_idx,2),1,size(d_idx,4)]);
e_idx = (q == d_idx);
e_idx = e_idx.*d_pstrat;

%Plot for PSSS correctness
figure;
[grid_cfo, grid_sinr] = meshgrid(sinr,cfo_set);
surf(grid_cfo,grid_sinr,sum(sum(e_idx,4),3)./sum(sum(d_pstrat,4),3));
colorbar;
zlabel('Probability'); ylabel('CFO Offset [Hz]'); xlabel('SINR [dB]');
xlim([min(sinr),max(sinr)]); ylim([min(cfo_set),max(cfo_set)]);
zlim([0 ,1]);

%Plot for symbol correctness
figure;
surf(grid_cfo,grid_sinr,sum(sum(d_pstrat,4),3)/(size(d_pstrat,4)*size(d_pstrat,3)));
colorbar;
zlabel('Probability'); ylabel('CFO Offset [Hz]'); xlabel('SINR [dB]');
xlim([min(sinr),max(sinr)]); ylim([min(cfo_set),max(cfo_set)]);
xlim([min(sinr),max(sinr)]); ylim([min(cfo_set),max(cfo_set)]);
zlim([0.8 , 1])
%% Probabilities for Sample Offset

s_off = squeeze(d_results(:,:,:,:,3)); %grab estimated offset index
s_off = s_off.*e_idx; %only grab the correct ones
s_off = reshape(s_off,[size(s_off,1),size(s_off,2),size(s_off,3)*size(s_off,4)]);
s_off = reshape(permute(s_off,[1 3 2]),[size(s_off,1)*size(s_off,3),size(s_off,2)]);

s_off_cnt = zeros(nfft+1,size(s_off,2));
for ii = 1:size(s_off,2)
    [s_off_cnt(:,ii), aa_bin] = hist(s_off(:,ii),0:128);
end

s_off_cnt = s_off_cnt(2:end,:);
s_off_cnt = circshift(s_off_cnt,-4);

s_off_culcnt = cumsum(s_off_cnt,1);
bb_cnt2 = bsxfun(ew_divide,s_off_culcnt,s_off_culcnt(end,:));
```

```
[grid_offset, grid_sinr]  = meshgrid(sinr,-63:64);
surf(grid_offset,grid_sinr,s_off_cnt)

plot(-63:64,bb_cnt2(:,[3 7 11 15 19]));
axis tight; ylabel('Probability'); xlabel('Sample Offset');
legend('SNR: 0 dB','SNR: -4 dB','SNR: -8 dB','SNR: -12 dB','SNR: -16 dB')
```

B.4 UE Class & Link Layer Simulations

```
classdef node

    properties
        %simulated probabilities
        c_offset;
        c_psss;
        c_symbol;
        thres_sym;

        %node state
        id;
        sync_source;
        slot;

        %current node params
        t_symbol_offset;
        t_sample_offset;

        %estimated params
        h_symbol_offset;
        h_sample_offset;

        sinr;
    end

    methods
```

```
%Constructor
function obj = node(id)
    %load probabilities
    load nodeconstructor.mat
    obj.c_offset = c_offset;
    obj.c_psss = c_psss;
    obj.c_symbol = c_symbol;
    obj.sinr = 2:-1:-16;
    obj.thres_sym = 2;

    obj.id = id; %global node id (unique)
    obj.slot = randi(4);
   obj.t_sample_offset = randi([-64 63]); %random sample offset
  obj.t_symbol_offset = randi([-700 699]); %random symbol offset
end

%update using collected estimates
function obj = update(obj)

    N = numel(obj.h_symbol_offset)+obj.sync_source;
    ot_sym = obj.t_symbol_offset;
    obj.t_symbol_offset = round((sum(obj.h_symbol_offset)+...
        obj.sync_source*obj.t_symbol_offset)/N);
    N = numel(obj.h_sample_offset)+obj.sync_source;
    obj.t_sample_offset = round((sum(obj.h_sample_offset)+...
        obj.sync_source*obj.t_sample_offset)/N);
    obj.t_sample_offset = ((sum(obj.h_sample_offset)+...
        obj.sync_source*obj.t_sample_offset)/N);
    if (~obj.sync_source)
       if ((abs(ot_sym - obj.t_symbol_offset) < obj.thres_sym))
            obj.sync_source = 1;
        end
    end

    obj.h_symbol_offset = [];
    obj.h_sample_offset = [];
end

%process an individual estimate
function obj = process(obj, SNR, true_sym, true_offset)
```

```
            if (SNR > max(obj.sinr))
                SNR = max(obj.sinr);
            elseif (SNR < min(obj.sinr))
                SNR = min(obj.sinr);
            end

            p_symbol = interp1(obj.sinr,obj.c_symbol,SNR);
            p_psss = interp1(obj.sinr,obj.c_symbol,SNR);
            p_offset1 = interp1(obj.sinr,2*obj.c_offset(64,:),SNR);
            p_offset0 = interp1(obj.sinr,obj.c_offset(65,:),SNR);

            choose_psss = rand < (p_symbol*p_psss);
            if (choose_psss)
                obj.h_symbol_offset = [obj.h_symbol_offset; true_sym];
                offset_choice = rand;
                if (offset_choice < p_offset1)
                    obj.h_sample_offset = [obj.h_sample_offset; ...
                        (true_offset + 2*(rand > 0.5) -1)];
                elseif (offset_choice < (p_offset0 + p_offset1))
                    obj.h_sample_offset = [obj.h_sample_offset; ...
                        true_offset];
                end
            end

        end

    end
end
```

```
addpath('helpers')
num_nodes = 7;
sym_per_slot = 7;
sym_per_100 = 1400;

for ii = 1:num_nodes
    q(ii) = node(ii);
```

```
    q(ii).sync_source = 1;
end
q(1).sync_source = 0;

%slot choices init
r_slot = randperm(4);
node_num = [2 3 4];
r_so = randi([-64 63]);
r_sao = randi([-700 699]);

for ii = 1:numel(node_num)
    q(node_num(ii)).slot = r_slot(ii);
    q(node_num(ii)).t_symbol_offset = r_sao;
    q(node_num(ii)).t_sample_offset = r_so + randi([-2 2]);
end
r_slot = randperm(4);
node_num = [5 6 7];
r_so = randi([-64 63]);
r_sao = randi([-sym_per_100/2 (sym_per_100/2)-1]);
for ii = 1:numel(node_num)
    q(node_num(ii)).slot = r_slot(ii);
    q(node_num(ii)).t_symbol_offset = r_sao;
    q(node_num(ii)).t_sample_offset = r_so+ randi([-2 2]);
end

k = 5;
d_l = 10;


d0 = -20*log10(sqrt(k^2 + (d_l/2)^2));
d1 = -20*log10(k+d_l/2*sqrt(3));
d = -20*log10(d_l);

ds = [Inf,d0 ,d0,d1,d0,d0,d1;
       d0,Inf,d,d,-Inf,-Inf,-Inf;
       d0, d,Inf,d,-Inf,-Inf,-Inf;
       d1, d, d, Inf,-Inf,-Inf,-Inf;
       d0,-Inf,-Inf,-Inf,Inf,d,d;
       d0,-Inf,-Inf,-Inf,d,Inf,d;
       d1, -Inf,-Inf,-Inf,d,d,Inf];
```

```
ds = ds - max([d0, d1, d]);


cycle_per_update = 2;
num_updates = 25;
end_offset = zeros(num_updates,3,numel(q));
end_offset_start = zeros(1,3,numel(q));
for jj = 1:numel(q)
   end_offset_start(1,:,jj) = [q(jj).t_symbol_offset, q(jj).t_sample_offset, q(jj).sync

end

for ii = 1:num_updates
    SNR = (ds~=-Inf)*2; %(from X to Y)
    SNR(SNR == 0) = -Inf;
    offset_gs = zeros(numel(q),1);
    for yy = 1:numel(q)
     offset_mod = [0 sym_per_slot sym_per_100 sym_per_100+sym_per_slot];
     offset_gs(yy) = mod(q(yy).t_symbol_offset + offset_mod(q(yy).slot),sym_per_100*2)
    end

    for yy = 1:numel(q)
        if(sum(ismember(offset_gs,offset_gs(yy))) > 1)
            ni = 0;
            for zz = 1:numel(q)
                if (yy == zz)
                    continue;
                end

                s = db2mag(ds(yy,zz));
                ni= ismember(offset_gs,offset_gs(yy));
                ni(yy) = 0;
                ni = sum(db2mag(ds(ni,zz)));
                SNR(yy,zz) = mag2db(s/ni);
            end
        end
    end


    for cycle = 1:cycle_per_update
```

```
        for jj = 1:numel(q)
            for kk = 1:numel(q)
                if (jj == kk)
                    continue
                end
                if (q(kk).sync_source ~= 1)
                    continue;
                end
                if (SNR(jj,kk) < -1000) %i.e. if -infinity
                    continue;
                end
           q(jj) = q(jj).process(SNR(kk,jj),q(kk).t_symbol_offset, q(kk).t_sample_offs
            end
        end
    end

    for jj = 1:numel(q)
        q(jj) = q(jj).update;
    end

    for jj = 1:numel(q)
     end_offset(ii,:,jj) = [q(jj).t_symbol_offset, q(jj).t_sample_offset, q(jj).sync_s
    end



    if (mod(ii,1) == 0)
    end


end



bx = [-100 100];
loc = (bx(2)-bx(1))* rand([2 num_nodes]) + bx(1);
figure; hold on; scatter(loc(1,:),loc(2,:)); scatter(loc(1,node_num),loc(2,node_num),

ds = zeros(num_nodes);
for ii = 1:numel(q)
```

```
    for jj = 1:numel(q)
     ds(ii,jj) = -20*log10(sqrt((loc(1,ii)-loc(1,jj))^2 + (loc(2,ii)-loc(2,jj))^2));
    end
end
```

References

[1] 3GPP. High Speed Downlink Packet Access: UE Radio Transmission and Reception (FDD) (Release 5). TR 25.890, 3rd Generation Partnership Project (3GPP), 2002. URL www.3gpp.org.

[2] 3GPP. Evolved Universal Terrestrial Radio Access (E-UTRA); Physical Channels and Modulation (Release 12). TS 36.211, 3rd Generation Partnership Project (3GPP), 2015. URL www.3gpp.org.

[3] 3GPP. Evolved Universal Terrestrial Radio Access (E-UTRA); Physical Layer Procedures (Release 12). TS 36.213, 3rd Generation Partnership Project (3GPP), 2015. URL www.3gpp.org.

[4] 3GPP. Evolved Universal Terrestrial Radio Access (E-UTRA); Overall description (Release 12). TS 36.300, 3rd Generation Partnership Project (3GPP), 2015. URL www.3gpp.org.

[5] 3GPP. Release 12, June 1 2016. URL http://www.3gpp.org/specifications/releases/68-release-12.

[6] WiFi Alliance. Wi-fi, June 1 2016. URL http://www.wi-fi.org/.

[7] S Beyme and C Leung. Efficient computation of dft of zadoff-chu sequences. *Electronics letters*, 45(9):461–463, 2009.

[8] A.G. Dabak and E.N. Onggosanusi. Low-complexity primary synchronization sequences, February 17 2010. URL https://www.google.com/patents/EP2153572A1?cl=en. EP Patent App. EP20,080,747,189.

[9] EN ETSI. 300 744 v1. 5.1,". *Digital video broadcasting (DVB)*, pages 2004–2011, 2004.

[10] Solomon W Golomb et al. *Shift register sequences*. Aegean Park Press, 1982.

[11] Qualcomm Incorporated. Signal Design for D2D Synchronization. Technical Report R1-142964, 3rd Generation Partnership Project (3GPP), 2014.

[12] ITU. Asymmetric digital subscriber line 2 transceivers. G 992.5, International Telecommunication Union, 2005. URL `www.itu.int`.

[13] Andreas F Molisch. *Wireless communications*. John Wiley & Sons, 2007.

[14] Branislav M Popovic. Generalized chirp-like polyphase sequences with optimum correlation properties. *IEEE Transactions on Information Theory*, 38 (4):1406–1409, 1992.

[15] Dennis Schaeffer. Power De-rating Reduction using DFT-SOFDM Reverse Link. Technical Report C30-20060424-028, 3rd Generation Partnership Project (3GPP), 2006.

[16] Rohde Schwarz. Device to Device Communication in LTE. White Paper 1MA264, Rohde Schwarz, 2015. URL `www.3gpp.org`.

[17] Stefania Sesia, Issam Toufik, and Matthew Baker. *LTE-the UMTS long term evolution*. Wiley Online Library, 2015.

[18] Bernard Sklar. *Digital communications*, volume 2. Prentice Hall NJ, 2001.

[19] Philipp Sommer and Roger Wattenhofer. Gradient clock synchronization in wireless sensor networks. In *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*, pages 37–48. IEEE Computer Society, 2009.

[20] Keysight Technologies. LTE Physical Layer Overview. `http://rfmw.em.keysight.com/wireless/helpfiles/89600B/WebHelp/Subsystems/lte/content/lte_overview.htm`, 2016.

[21] Wikipedia, the free encyclopedia. Ofdm transmitter ideal, 2016. URL `https://commons.wikimedia.org/wiki/File:OFDM_transmitter_ideal.png`.

[22] Xinzhou Wu, Saurabh Tavildar, Sanjay Shakkottai, Tom Richardson, Junyi Li, Rajiv Laroia, and Aleksandar Jovicic. Flashlinq: A synchronous distributed scheduler for peer-to-peer ad hoc networks. In *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, pages 514–521. IEEE, 2010.