LOW LATENCY CDMA-BASED PROTOCOL TO SUPPORT IOT TRAFFIC IN 5G

BY SIDDARTH MATHUR

A thesis submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Prof. Dipankar Raychaudhuri

and approved by

_____

_____

_____

New Brunswick, New Jersey

October, 2017

ABSTRACT OF THE THESIS

Low Latency CDMA-Based Protocol to Support IoT Traffic in 5G

by Siddarth Mathur

Thesis Director: Prof. Dipankar Raychaudhuri

The upcoming 5th generation mobile network architecture is envisioned to deploy massive Internet-of-Things (IoTs) devices with a variety of traffic patterns. These devices will often transmit short sporadic messages, which are not well suited to the connection-oriented modes associated with legacy 3GPP networks resulting in high service latency and excessive control overhead. This thesis presents the design of a low latency MAC (Medium Access Layer) / PHY (Physical Layer) protocol for emerging Internet of Things (IoT) devices that require low access delay. The goal is to operate in the same band as current LTE, thus not requiring any separate channel allocation, while maintaining backward compatibility with the current LTE system. The physical layer access is achieved using an underlay CDMA-based low power transmission scheme, which operates in the same frequency range as the LTE's uplink/downlink frequencies. The MAC layer is designed for low access latency by transmitting small sized data in a random access mode as it becomes available, eliminating the need to setup a connection.

A proof of concept prototype was developed to demonstrate the feasibility of the proposed design and the performance of the CDMA system and in presence of LTE. The CDMA based transmission was prototyped using the Software Defined Radio (SDR)

platform (USRP B210/X310) and the code is written in C and C++. The LTE transmission is enabled using the OpenAirInterface (OAI) platform, which is an open sourced LTE implementation for Software Defined Radios. The performance of CDMA is studied with varying the spreading code length, message size, delay between transmitted packets, Signal to Noise Ratio (SNR). The CDMA based system is studied independently as well as in the presence of an ongoing LTE transmission. The results demonstrate that underlay burst CDMA transmissions for IoTs are capable of providing lower latency compared to LTE.

Acknowledgements

I would like to sincerely express my deepest gratitude to my adviser Dr. Dipankar Raychaudhuri for his constant support, sound advice and invaluable guidance to shape my thesis. I would like to thank my mentor Prof. Dola Saha. Her constant support and direction was invaluable for my work and it was a great learning experience working with her. I would also like to thank my colleagues at Huawei, Santa Clara to support me in my time there working on this project and gave me some great advice.

I would like to thank the rest of my thesis committee members, Dr. Zoran Gajic and Dr. Predrag Spasojevic. Finally, I would like to thank my friends at WINLAB and my family for their unwavering support.

Table of Contents

List of Figures

Chapter 1

Introduction

Global data traffic will increase sevenfold between 2016 and 2021. Mobile data traffic will grow at a rate of 47 percent from 2016 to 2021 [5]. In 2016, 4G carried 69 percent of the total mobile traffic and is expected to represent over 79 percent of all mobile traffic by 2021. Smartphones have been a part of this trend for quite a while now, but in near future, the majority increases in the number of devices will be due to the emerging Machine to Machine (M2M) communication systems, also termed as Internet of Things (IoT). M2M connections were initially forecasted to be about 50 billion by 2020 [5, 6]. This estimate although seeming farfetched now, the number is still expected to be in the range of 20B to 30B connected devices [7, 8]. This recent evolution introduces a great variety of network-enabled objects which interact with each other and provide a broad spectrum of new services. Connectivity is now reaching home devices, cars, industrial automation, etc. and wireless mobile networks are one of the main platforms for these M2M systems .

5G is the next phase of mobile communication and is expected to have significantly higher bandwidth (Gigabit speeds), broader coverage and ultra low latency. The current fourth generation of mobile networks, the Long Term Evolution (LTE) has been designed to enhance the capacity in order to provide support to a large number of connected devices and is expected to be the main driver of the emergence of the IoT on cellular networks. These mobile networks are designed and optimized to transport human-originated traffic. The traffic characteristics of many IoT applications which are substantially different than user traffic from smartphones and tablets, are known to result in network resource utilization inefficiencies. Machine type communication involves a potentially large number of communicating terminals with, to a large extent,

little traffic per terminal transmitted between them [9].

Considering the aforementioned traffic requirements, there is a need to design a network with low access latency and less control overhead. This can by achieved by using a random access protocol. Most of the proposed IoT systems require additional spectrum allocation, underlining the need to coexist with the current cellular network in the same frequency band. Our design is a MAC/Physical layer solution for low power, low bitrate and static devices that require low access delay (possibly as low as 1-2 ms) and long range for communication. We propose a CDMA-based low power transmission for simultaneous channel access of IoT devices along with LTE User Equipments (UE). In such cases, the IoT devices can operate at low Signal-to-Noise Ratio (SNR) for low bit-rate data transmission [10]. There has been work in the community with SIGFOX [11], which is a Low Throughput and Ultra Narrow-Band(UNB) Modulation system limiting the maximum size to 12 bytes of information on 200 Hz bandwidth. NB-IoT [12] is another standard for MTC, using 180 KHz of bandwidth using OFDMA/SC-FDMA. However, it targets latency insensitive applications and targets a latency of 10 *seconds* for emergency messages. LoRa [13] Alliance is yet another option which uses proprietary chirp-based spread spectrum technique developed by Semtech.

Chapter 2 provides a brief overview of the LTE Network Architecture and provides us with the motivation behind the proposed design in detail. Delays in LTE and CDMA are discussed to justify the use of underlay CDMA. Capacity analysis of standalone CDMA based IoT network and CDMA in presence of LTE is done in detail. Chapter 3 builds on this and provides a detailed description of the underlay CDMA technique and why it is a feasible option to use. Chapter 4 presents a proof of concept prototype to demonstrate the functionality of the prototyped CDMA on USRPs (Universal Software Radio Peripherals). The experimental evaluations of the standalone CDMA and in presence of overlay LTE transmissions are presented. Finally, the related work and Conclusion are presented in Chapter 5 and Chapter 6 respectively.

Chapter 2

Background and Motivation

2.1 LTE Architecture Overview

Long Term Evolution (LTE) is a 4G mobile communications standard for high speed wireless communication majority between mobile phones and data terminals. It is a standard by 3GPP (3rd Generation Partnership Project) [14]. LTE Release 8 supports peak data rates up to 326 Mbps on the downlink and 86 Mbps on the uplink with a 20 GHz channel and 4x4 MIMO (Multiple Input, Multiple Output Antennas). It provides duplexing in both Frequency (Frequency Division Duplexing) and Time (Time Domain Duplexing). The network architecture of LTE consists of two blocks, i.e. the E-UTRAN (Evolved UTRAN) and the evolved packet core (EPC). The E-UTRAN consists of the eNodeB (Base Station) and the UE (User Equipment). The Evolved Packet core consists of the Mobility Management Entity (MME), Home Subscriber Server (HSS), S-GW (Service Gateway), P-GW (PDN Gateway) and the Policy Control and Charging Rules Function (PCRF). Figure 2.1 shows the overall network architecture of LTE. It includes the different network elements and the standard interfaces between them. In this section. we look at the E-UTRAN and the Core Network (CN) for inefficiencies that could be ill-suited for the target IoT applications.

Our design objective is to reduce the network latency in LTE for short messages generated by IoTs, and to reduce the overhead generated in the network for these devices. Latency in LTE consists of :

- Control plane latency
- User plane latency

Control plane latency deals with signaling and controlling functions and is measured

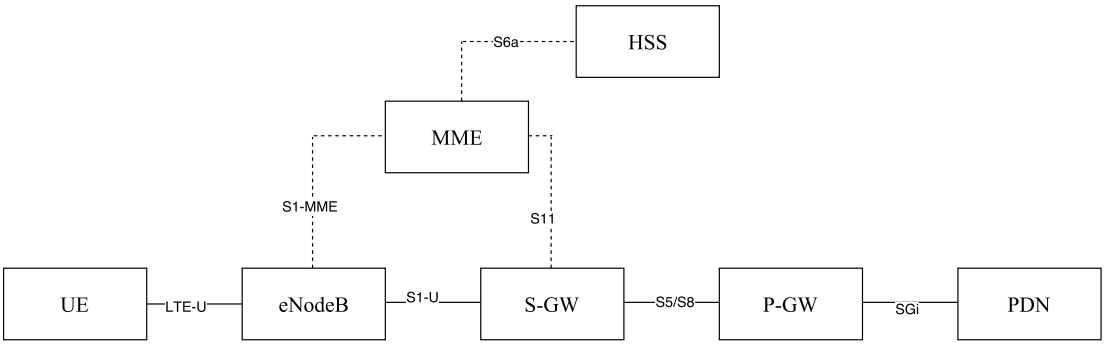


Figure 2.1: LTE Architecture Diagram

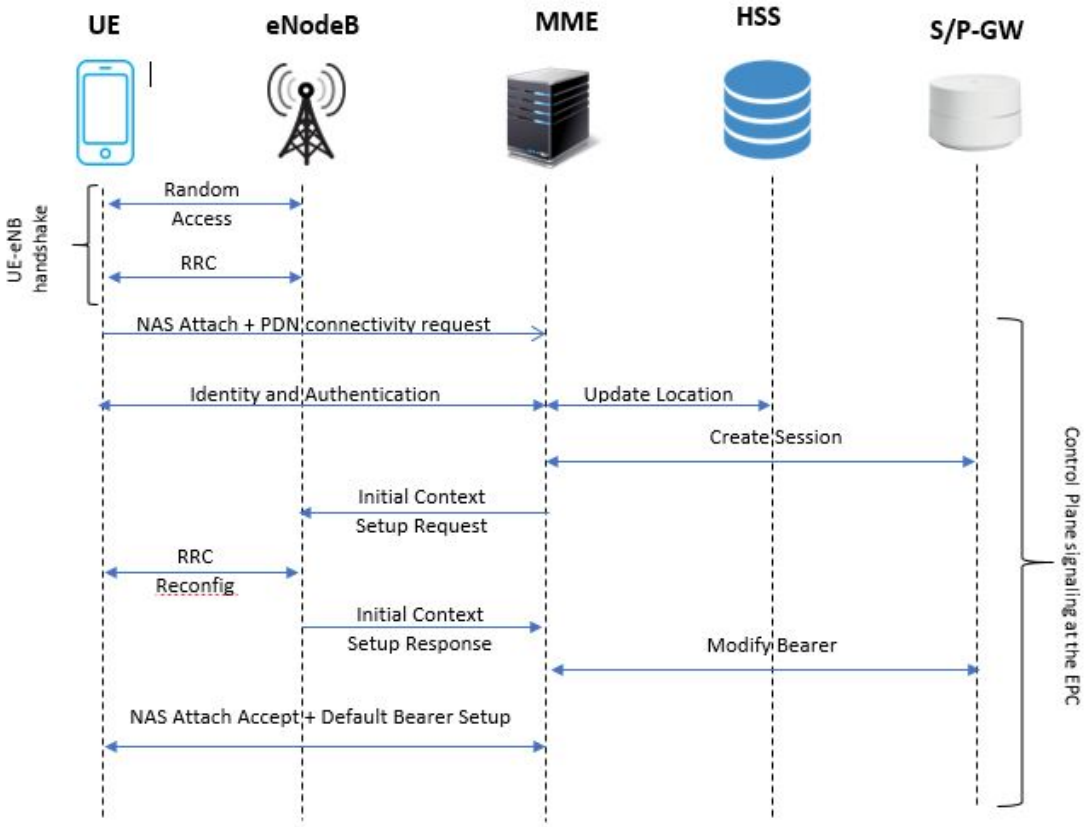


Figure 2.2: Control Plane Signaling (User Equipment Attach Procedure)

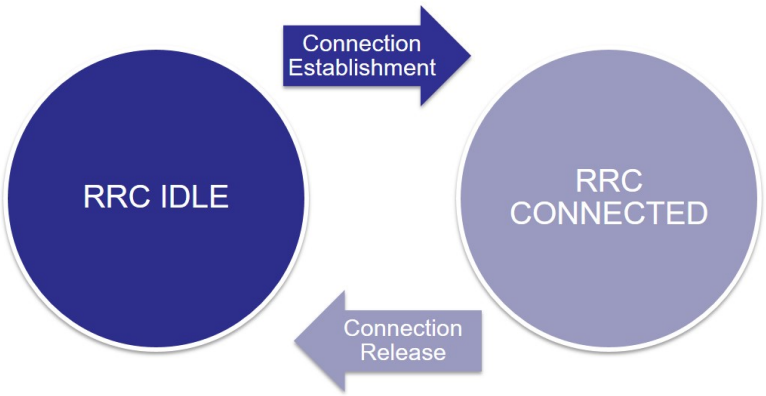


Figure 2.3: RRC State transition in LTE

as the transit time from an idle state (Radio resource control (RRC) Idle) to an active state (RRC Connected) for the UE. User Plane latency deals with the actual data transmission, and is defined as transit time between a packet available at IP layer of the UE and the availability at the IP Layer of the eNodeB. Our focus is mainly on the control plane since it is the biggest piece in the network latency.

A UE looking to access the network, has to perform an attach procedure to the network. If the UE has data to transmit, it initiates a connection by means of the Random Access Procedure in which it sets up the RRC (Radio Resource Control) Connection. Non Access Stratum (NAS) (UE and MME) security, Access Stratum (AS) (UE and eNB) security and mutual authentication procedures are then performed and a bearer is finally established through the Evolved Packet Core (EPC) in order to send and receive user traffic. This accounts up to 30 % of control plane signaling overhead.

A device goes to the RRC Idle state if it has been inactive for 10 seconds and for the next transmission, the device needs to reestablish the radio bearer as seen in Figure 2.3. The idle to connected latency is around 60ms [15], which is very high compared to WiFi (often ¡1ms). This increases control plane overhead within the network. Figure 2.4 shows the cumulative control overhead, which is extremely high when the packet size is very small. M2M devices will have low throughput and the signaling load is

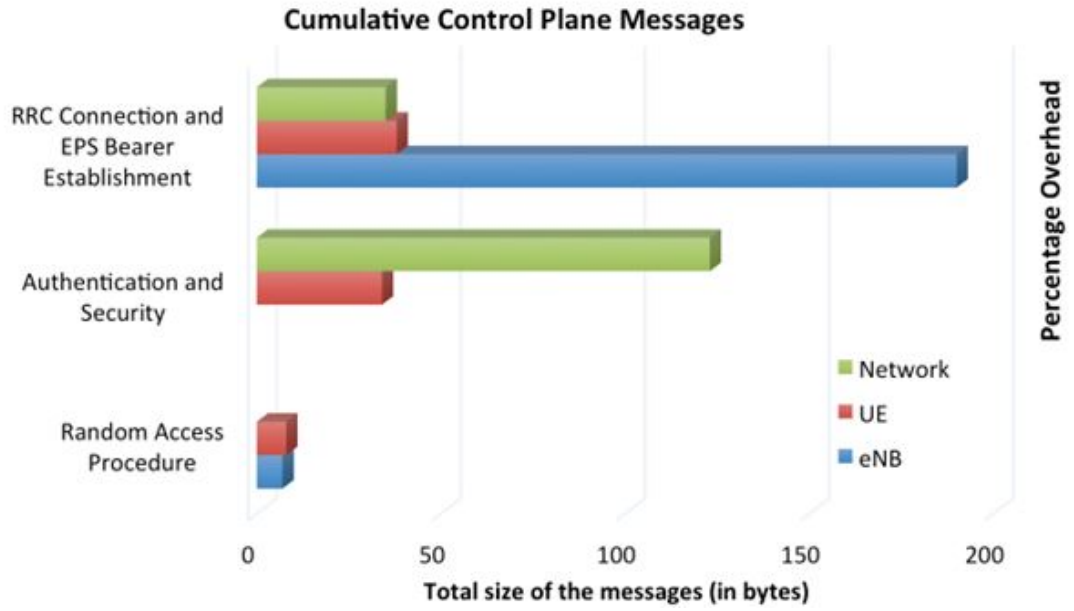


Figure 2.4: Cumulative Overhead in Control Plane for Attach Procedure in LTE [1]

really high due to the control overhead. Now, the two major issues in using the existing 4G net- work for IoT traffic are:

- High latency for the attach procedure .
- High overhead in control plane.

As can be seen in Figure 2.4, higher latency is due to the wait time at UE for random access and the time to perform multiple steps of attach procedure. Figure 2.5 shows the percentage overhead for message transmissions by an idle UE. The overhead can be as high as 4500% for a 1 byte message, when the UE is in idle mode and it has to re-establish the connection to the network.

We will minimize the waiting time by using an underlay CDMA based transmission with low access latency for burst IoT traffic without having the need to over provision the allocated spectrum.

This motivates us to propose an underlay CDMA-based low power IoT transmission for the shared band operation of IoT and LTE devices. With the proposed underlay CDMA-based MAC and Physical layer solution, we achieve low latency, short-message and long range communication required for low-power IoT devices.

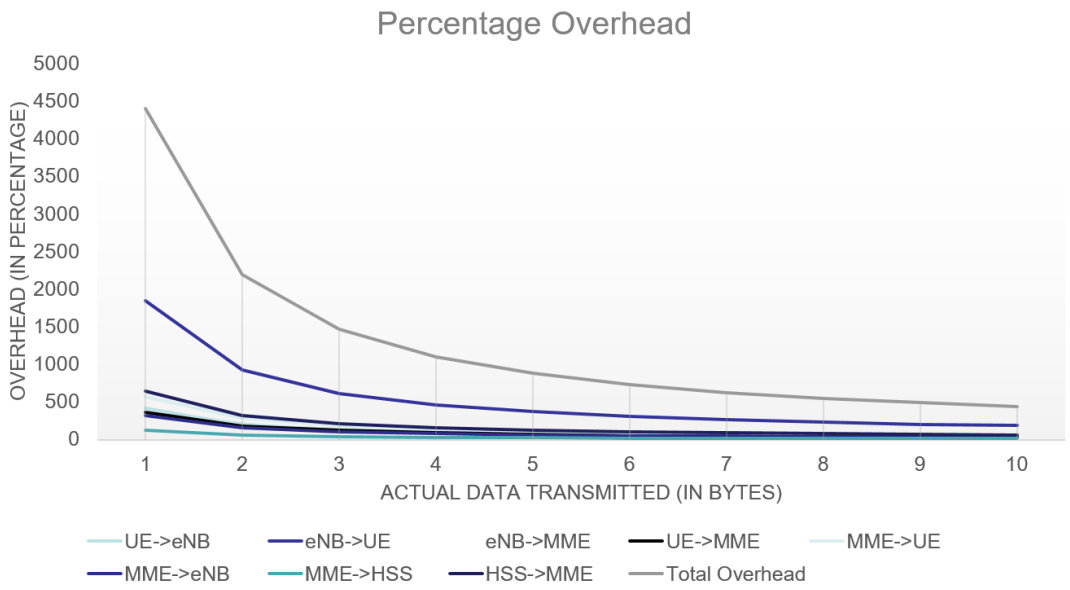


Figure 2.5: Percentage Overhead for transmissions of different sizes in idle mode [1]

2.2 Code Division Multiple Access

Code Division Multiple Access or CDMA is a spread spectrum, multiple access technique. It uses a spread spectrum technique to spread the bandwidth of the data uniformly over the available frequency band. It allows numerous signals or clients to occupy a single transmission channel and optimizing the use of available bandwidth. The spreading is achieved by codes, which are independent of the data. These codes are pseudo-random There are two types of spread spectrum techniques :

- *Frequency Hopping Spread Spectrum (FHSS)*

  The transmitters hops between different frequencies repeatedly within a specified channel bandwidth, using a pseudo-random sequence known to both sender and receiver. Data is transmitted on a sequence of different narrow band frequencies specified by the PN sequence.

- *Direct sequence Spread Spectrum (DSSS)*

  The sender uses a PN sequence code or also known as "chipping code" to spread the data over a wide bandwidth. The other senders use different chipping codes and transmit on the same frequency bandwidth. These chipping sequences are

orthogonal. They give high auto-correlation output and zero cross-correlation with other chipping sequences.

Walsh-Hadamard codes are used as chip sequences in our work. The Walsh-Hadamard codes encode n bits into 2n bits.

2.2.1 CDMA Capacity

In CDMA [16], the information-bearing baseband signal, $s(t)$, is multiplied by the spreading code, $c(t)$. Let us assume that T_b is the bit interval of $s(t)$ with the information rate of $R = 1/T_b$ bits/sec and T_c is the pulse duration of $c(t)$. For such system, CDMA processing gain is given by

$$L_c = \frac{T_b}{T_c} = \frac{W}{R_b} \tag{2.1}$$

where $W \approx 1/T_c$ represents the total spread bandwidth. In the proposed underlay CDMA IoT network, we assume all uplink signals (IoT device-to-eNB) are received at the same power level S. For N IoT UEs, signal-to-interference-plus-noise-ratio (SINR) is given by [17]

$$\text{SINR} = \frac{\text{S}}{(\text{N}-1)\text{S} + \eta} \tag{2.2}$$

where η is the thermal noise, and E_b/N_0 can be obtained as

$$E_b/N_0 = \frac{S/R_b}{(N-1)S/W + \eta/W} = \frac{W/R_b}{(N-1) + \eta/S} \tag{2.3}$$

For a given value of E_b/N_0, required for the adequate performance of demodulation and decoding, the capacity of CDMA in terms of users it can support is

$$N = 1 + \frac{W/R_b}{E_b/N_0} - \frac{\eta}{S} \tag{2.4}$$

The probability of error for the CDMA system using BPSK modulation is expressed as

$$P_b = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \tag{2.5}$$

where $Q(x)$ is Q-function, E_b is the energy per bit, N_0 is the power spectral density of an equivalent broadband interference (generally, over bandwidth W) and E_b/N_0 is referred as the bit energy-to-noise density ratio.

System Parameter	**Specification**
CDMA bandwidth	1 MHz
Modulation	BPSK
Antenna configuration	UE: 1T1R
Operating frequency	2620.5 MHz
UE transmit power	23 dBm
Thermal noise density	-174 dBm/Hz
Cell site sector radius	600 m
Path loss model (for 2GHz < freq < 6GHz)	36.7 $\log_{10}$(dist[m]) + 22.7 + 26 $\log_{10}$(frq [GHz])

Table 2.1: Parameters for CDMA-based IoT network [2]

The capacity of the proposed CDMA IoT network for typical cellular deployment parameters (given in Table 2.1) is now evaluated [2]. As shown in Fig. 2.6, the capacity of CDMA IoT network, N, can be augmented by either increasing the processing gain (higher system complexity) or by compromising system performance in terms of bit error rate (BER), P_b. Assuming the BER system constraint $P_b = 10^{-3}$ (e.g. required P_b or better for digital voice transmission), the required E_b/N_0 is approximately 7 dB at $L_c = 64$ and for a powerful convolutional code. In a single CDMA channel, it can accommodate $N = 13$ IoT UEs simultaneously without affecting each other's performance. The total CDMA network capacity, $N_{\max}$ can be obtained as:

$$N_{\max} = \text{Capacity of a single CDMA channel * No. of CDMA channels} \tag{2.6}$$

For example, let's take a CDMA channel with bandwidth 1 MHz and $N = 13$ which can coexist with an LTE network using 20 MHz spectrum band. The maximum achievable CDMA IoT capacity in that case, with no LTE presence is 260 users. We look at some observations which affect the capacity of CDMA IoT network :

- **No requirement of guard bands**

  CDMA IoT network can avoid having guard bands between CDMA channels by using non-overlapping data spreading codes for adjacent frequencies. For example, a 64-bit Hadamard code forms 64 spreading code sequences. For $N = 13$, 4 groups of non-overlapping sequences can be formed. In this case, maximum channel separation is 4 MHz for $W = 1$ MHz minimizing the interference between adjacent

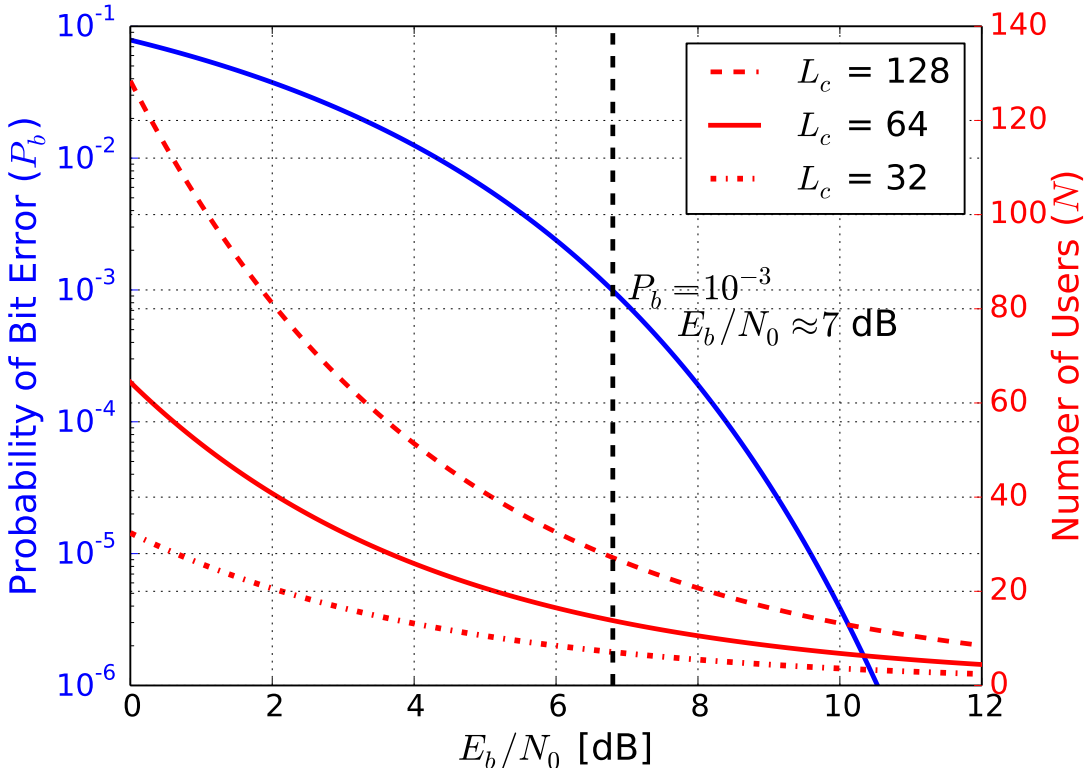


Figure 2.6: Bit error rate (P_b) and capacity of CDMA IoT network (N) as a function of E_b/N_0 and processing gain (L_c) for received power S at distance = 600 m [2]

channels.

- **Robust performance in multi-cell scenario**

  In the conservative approach of limited spectrum band with reuse factor 1, interference between CDMA UEs in adjacent cells can be mitigated by CDMA code and channel diversity. For example, for 20 CDMA channels with $W = 1$ MHz in 20 MHz band and 4 non-overlapping spreading sequence groups, code-channel diversity has a reuse factor of 80. This assures that IoT UEs in adjacent cells with the same channel and spreading code are at least apart by inter-eNB distance causing negligible interference to each other.

2.3 Comparison of Access Delay in LTE and Underlay CDMA

As seen in section 2.1, the control plane latency LTE is a hindrance for applications requiring low latency access or transmission times. Thus, control plane latency, i.e. Idle UE to Connected UE, can be divided into two parts. The initial being the access delay and the other being the transmission delay. *Access delay* is the delay an average idle UE experiences from the time it has a message to be sent to the time it actually transmits it to the network. In case of LTE, the access delay is the major part of the control plane latency. Transmission delay in LTE is minimal due to OFDM, which allows high data rate. For underlay CDMA, it is quite the opposite. The time to get access to the network is minimal, since it is a random access technique. Whereas, the transmission time is high depending on the spreading factor used and the size of the message. These delays in LTE and CDMA are discussed and the use of underlay CDMA is justified.

2.3.1 Latency in LTE

2.3.1.1 Transmission Delay in LTE

The focus is on IoT uplink transmissions and thus, we look at transmission times for LTE uplink in this section. The transmission times in LTE are extremely low due to multiple carriers in OFDM and higher orders of modulation used. The transmissions over only the Physical Uplink Shared Channel (PUSCH) are considered, which is the data channel. The approximate throughput in the PUSCH is calculated and the transmission time is derived from the throughput. There are three factors that control LTE throughput:

- *Bandwidth*

  The bandwidth is acquired by the UE from the Master Information Block (MIB), transmitted in the Physical Broadcast Channel (PBCH). The total number of resource blocks allocated for uplink, N_{PRB} is obtained from the bandwidth.

- *Modulation and Coding Scheme Index(I_{MCS}) and the redundancy version (RV)*

  This index determines the modulation scheme and the code rate to be used. UE

MCS Index I_{MCS}	Modulation Order Q'_m	TBS Index I_{TBS}	Redundancy Version rv_{idx}
0	2	0	0
1	2	1	0
2	2	2	0
3	2	3	0
4	2	4	0
5	2	5	0
6	2	6	0
7	2	7	0
8	2	8	0
9	2	9	0
10	2	10	0
11	4	10	0
12	4	11	0
13	4	12	0
14	4	13	0
15	4	14	0
16	4	15	0
17	4	16	0
18	4	17	0
19	4	18	0
20	4	19	0
21	6	19	0
22	6	20	0
23	6	21	0
24	6	22	0
25	6	23	0
26	6	24	0
27	6	25	0
28	6	26	0
29	reserved		1
30			2
31			3

Figure 2.7: Modulation, TBS index and redundancy version table for PUSCH [3]

would read the 5-bit index, I_{MCS} from the Downlink Channel Indicator(DCI) format 0 message in the PDCCH [3].

- *Transport Block Size Index(I_{TBS})* For $0 \leq I_{MCS} \leq 28$, the UE would determine the I_{TBS} in Table 8.6.1-1 in TS 36.213 [3] which can be seen in Figure 2.7.

The maximum throughput depends on the I_{TBS} and N_{PRB}. These two parameters determine the transport block size, which is the maximum number of bits that can be transmitted on the uplink in 1 TTI (= 1 millisecond). This value is obtained from the Table 7.1.7.2.1-1 in [3]. For example, considering 5 MHz bandwidth (= 25 Resource Blocks) and I_{MCS} to be 9 which uses QPSK modulation. I_{TBS} is 9 and TBS is 4008 bits as seen in Figure 2.8. Thus, the throughput is 4.008 Mbps. Assuming the user is using all resource blocks, transmission delay for a 200 byte(1600 bits) message is 0.4 milliseconds and for a 33 byte(254 bits) message is 0.065 milliseconds. The transmission latency is significantly lower than the access delay and thus can be ignored.

I_{TBS}	N_{PRB}									
	21	22	23	24	25	26	27	28	29	30
0	568	600	616	648	680	712	744	776	776	808
1	744	776	808	872	904	936	968	1000	1032	1064
2	936	968	1000	1064	1096	1160	1192	1256	1288	1320
3	1224	1256	1320	1384	1416	1480	1544	1608	1672	1736
4	1480	1544	1608	1736	1800	1864	1928	1992	2088	2152
5	1864	1928	2024	2088	2216	2280	2344	2472	2536	2664
6	2216	2280	2408	2472	2600	2728	2792	2984	2984	3112
7	2536	2664	2792	2984	3112	3240	3368	3368	3496	3624
8	2984	3112	3240	3368	3496	3624	3752	3880	4008	4264
9	3368	3496	3624	3752	4008	4136	4264	4392	4584	4776
10	3752	3880	4008	4264	4392	4584	4776	4968	5160	5352
11	4264	4392	4584	4776	4968	5352	5544	5736	5992	5992
12	4776	4968	5352	5544	5736	5992	6200	6456	6712	6712
13	5352	5736	5992	6200	6456	6712	6968	7224	7480	7736
14	5992	6200	6456	6968	7224	7480	7736	7992	8248	8504
15	6456	6712	6968	7224	7736	7992	8248	8504	8760	9144
16	6712	7224	7480	7736	7992	8504	8760	9144	9528	9912
17	7480	7992	8248	8760	9144	9528	9912	10296	10296	10680

Figure 2.8: Transport block size table [3]

2.3.1.2 Access Delay in LTE

An idle user waiting to get access from the eNodeB experiences some delay from when it has a packet to transmit and to the time it sends it across the air interface. The access delay is considered to be the time to get resources from the eNodeB and can be broadly divided into two steps:

1. **Initial Synchronization and Acquisition**

   The initial acquisition procedure has multiple steps involved which can be seen in Figure 2.9. The first step is the frequency scanning and selection. The UE tunes to all channels it supports and measures the Received Signal Strength Indicator (RSSI). The channels with high RSSI are determined. Primary Synchronization Signal (PSS) and Secondary Synchronization Signal (SSS) are used to achieve slot and radio frame synchronization. The Physical Cell Identity (PCI) is also retrieved. PSS and SSS are transmitted in the last and second last OFDM symbol of the first and sixth subframe. Next step is to decode the Master Information Block(MIB) which contains Bandwidth, System Frame Number (SFN), PHICH information. The MIB uses a fixed schedule with a periodicity of 40 ms and repetitions made within 40 ms. It is transmitted in the first 4 OFDM symbols of the second slot in the first subframe. The Physical Downlink Control Channel (PDCCH) is then monitored to decode DCI Format 1A which gives PDSCH information. System Information Blocks(SIBs) are transmitted in PDSCH.The

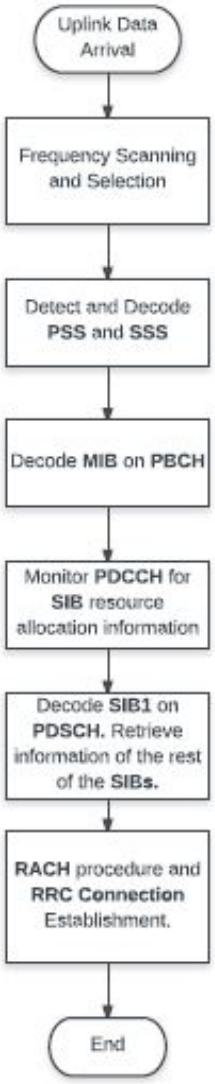


Figure 2.9: Initial Acquisition Procedure in LTE

SIB1 and SIB2 are vital and others are optional. SIB1 specifies the timing of remaining SI blocks, along with PLMN identity. It is transmitted in the subframe 5 of frame with $SFNmod2 = 0$. SIB2 contains RACH, PUSCH and PUCCH related parameters and idle mode paging configurations. The periodicity of SIB2 is given in SIB1, with minimum period 80 radio subframes and maximum period 512 subframes [18]. The best and worst times are calculated and can be seen in Figure 2.10. The minimum acquisition time is about 19.857 milliseconds and the worst acquisition time is as high as 5.13 seconds.

2. **Random Access Procedure (RAP)**

   Random access procedure is a procedure for the UE to gain access to resources after the initial cell acquisition. It is done over the Random Access Channel (RACH). There are two types of RAP, namely, Contention Based and Contention free. A preamble is sent by the UE to initiate the RAP and there are 64 preambles available. Contention based random access is a four-step procedure as is shown in Figure2.11. It starts with UE selecting one of the 64 RACH preambles and sending it to the eNodeB. The eNodeB sends a Random Access Response (RAR) message to UE on the DL-SCH(Downlink Shared Channel). This message contains uplink resource grant and then Msg3 (RRC Connection Request) is transmitted by the UE using UL-SCH(Uplink Shared Channel). This message contains UE's temporary identity and connection establishment cause. eNodeB responds with a RRC Connection Setup message(Msg4) which is a contention resolution message, which contains a new permanent identifier for the UE. Average delay to complete the RACH procedure as seen in Table II of [15] is $\sim 50ms$, which is a significant amount of delay.

For a large number of devices, unexpected bursty arrivals of access requests may result in severe collisions in RACH. Here, we look at an analytical model [4] to investigate the behavior of the RACH with bursty data arrival, as expected in case of IoTs. The contention-based random access is a slotted ALOHA based access mechanism, where each UE transmits a preamble in the first available random-access slot.

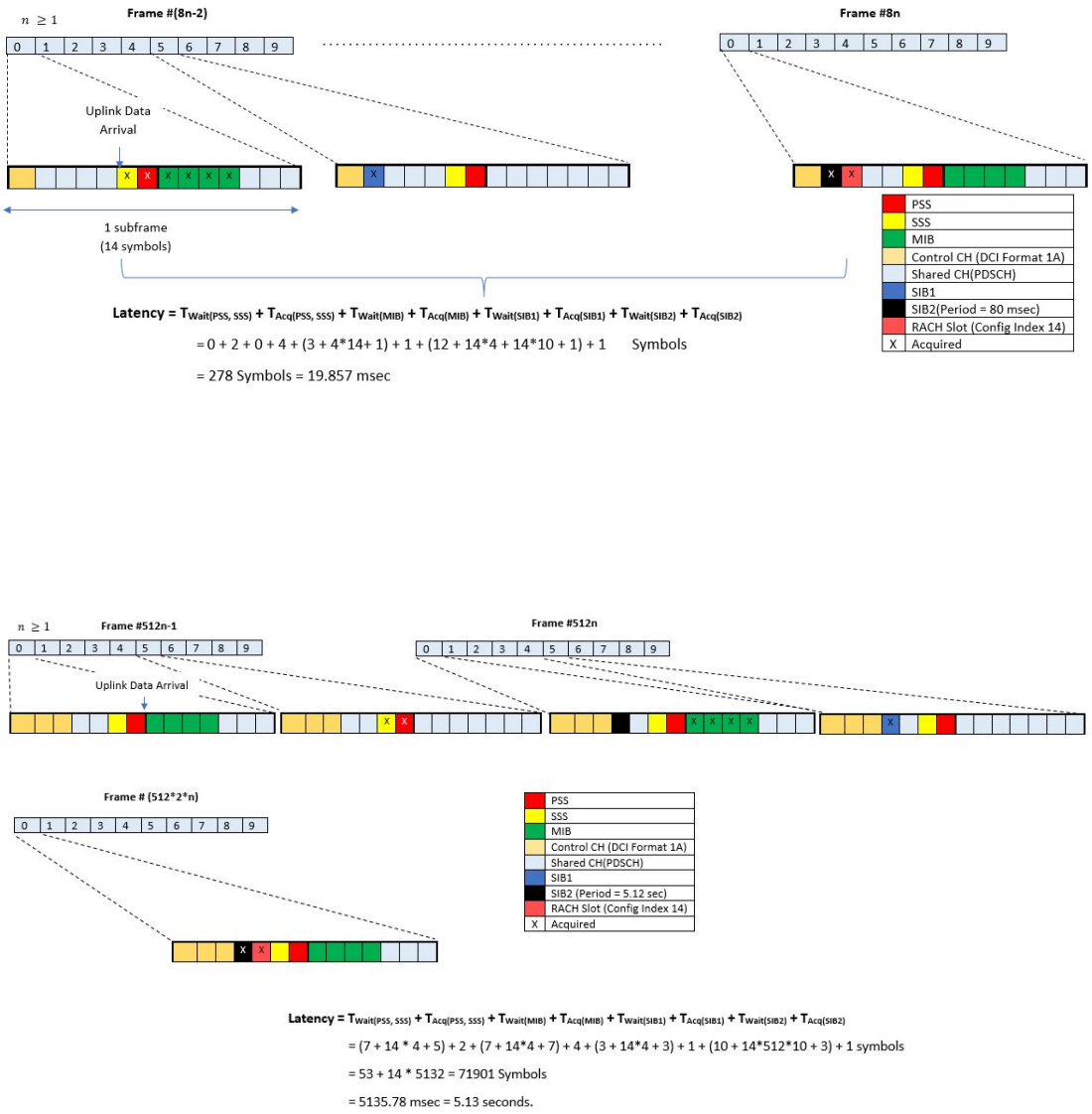
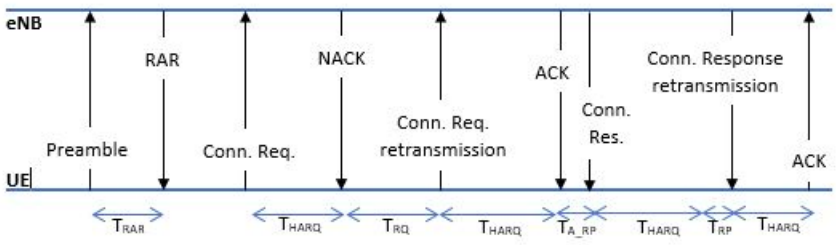


Figure 2.10: Best and Worst Initial Acquisition times for an idle UE in LTE

Figure 2.11: Contention Based Random Access Procedure [4]



NOTATION	MEANING
M	Number of arriving UEs during time interval T_P
T_P	Arrival period (sub-frames)
T_{RAR}	Processing time required by the eNB to detect the transmitted preambles (sub-frames)
T_{RQ}	Gap of *Connection Request* message retransmission (sub-frames)
T_{A_RP}	Gap of Monitor *Connection Response* message (sub-frames)
T_{RP}	Gap of *Connection Response* message retransmission (sub-frames)
T_{HARQ}	Time interval required for receiving HARQ ACK (sub-frames)
$p_{e,MSG}$	Error probability of the message transmission
I_R	Number of random-access slots
W_{BO}	Backoff window size (sub-frames)
T_{RA_REP}	Interval between two successive random-access slot (sub-frames)
N_{RAR}	Maximum number of RAR that can be carried in a response message
W_{RAR}	Length of the random-access response windows (sub-frames)
p_f	HARQ retransmission probability for a *Connection Request/Response* message
N_{HARQ}	Maximum number of HARQ transmissions for a *Connection Request/Response* message
p_n	Preamble detection probability of the n-th preamble transmission

Table 2.2: Basic parameters used for RACH modeling [4]

Average access delay for the successfully accessed UEs, $\overline{D_a}$, is the ratio between the total access delay for all successfully accessed UEs and the total number of successfully accessed UEs [4]. $\overline{D_a}$ is given by:

$$\overline{D_a} \approx \frac{\sum_{i=1}^{I_R} \sum_{n=1}^{N_{PTmax}} M_{i,S}[n]\overline{T_n}}{\sum_{i=1}^{I_R} \sum_{n=1}^{N_{PTmax}} M_{i,S}[n]} \tag{2.7}$$

where $\overline{T_n}$ is the average access delay of a successfully accessed UE that transmits exactly n preambles. This includes the time to transmit the first preamble, to re-transmit (n-1) preambles, eNB processing times and finishing the message(Msg3 and Msg4) transmissions. $M_{i,S}[n]$ represents the number of successful transmissions during random-access slot $i \geq 0$ by UEs engaged in their n-th transmission attempt.

$$\overline{T_n} \approx 1 + (n-1)\overline{T_W} + T_{RAR} + W_{RAR} + \overline{T_{MSG}} \tag{2.8}$$

where $\overline{T_W}$ is the average time a UE waits to perform backoff and re-transmit a preamble. $\overline{T_{MSG}}$ is the average message transmission time. Considering a case where the UE fails in the first preamble transmission in the 1st random=access slot and retransmits another preamble in the $(1+h)$th random-access slot. $\overline{T_W}$ is given by:

$$\overline{T_W} \approx \sum_{h=H_{min}}^{H_{max}} q_h h T_{RA_R EP} \tag{2.9}$$

where H_{min} and H_{max} are the minimal and maximal value of h. H_{min} and H_{max} occur when the backoff counter is zero and W_{BO}, respectively.

$\overline{T_{MSG}}$ can be obtained by considering that the message transmissions are completed by sending u HARQ transmissions of the *ConnectionRequest* message and receiving v HARQ transmissions of *ConnectionResponse* message. $\overline{T_{MSG}}$ in [4] is given by:

$$\overline{T_{MSG}} = \sum_{u=1}^{N_{HARQ}} \sum_{v=1}^{N_{HARQ}} p_f^{u+v-2}(1-p_f)^2[(u-1)T_{RQ}+(v-1)T_{RP}+(u+v)T_{HARQ}+T_{A\_RP}+1] \tag{2.10}$$

Using the equations above, the average access delay is calculated in [4] for two traffic models defined in 3GPP TR37.868. in [19]. As the number of arriving UEs during a time interval T_P are increased from 5000 to 30,000, the access delay varies between $\sim$ 28 ms to 65 ms. The arrival period, T_P is 10000 and 60000 for the two traffic models.

2.3.2 Latency in CDMA

2.3.2.1 Transmission Delay in CDMA

The delay in transmission is a significant part of the latency in CDMA. It is the time from when the packet is available at the Physical layer of the transmitter to the time when it is transmitted over the air. With respect to CDMA, there are a few parameters that we vary and study the delay. The parameters are :

- Bandwidth *(1 MHz - 20 MHz)*
- Modulation Scheme *(BPSK - 64QAM)*
- Message Size *(1 - 200 bytes)*
- Spreading factor *(32 - 512 bits)*

The IoT data traffic model as seen in the official 3GPP document can be seen in Table 2.3 [2] [20]. The average payload size of the IoT data traffic is retrieved

Figure 2.12: Transmission Delay (in ms) for a BPSK Modulated CDMA Transmission, varying spread/preamble code lengths

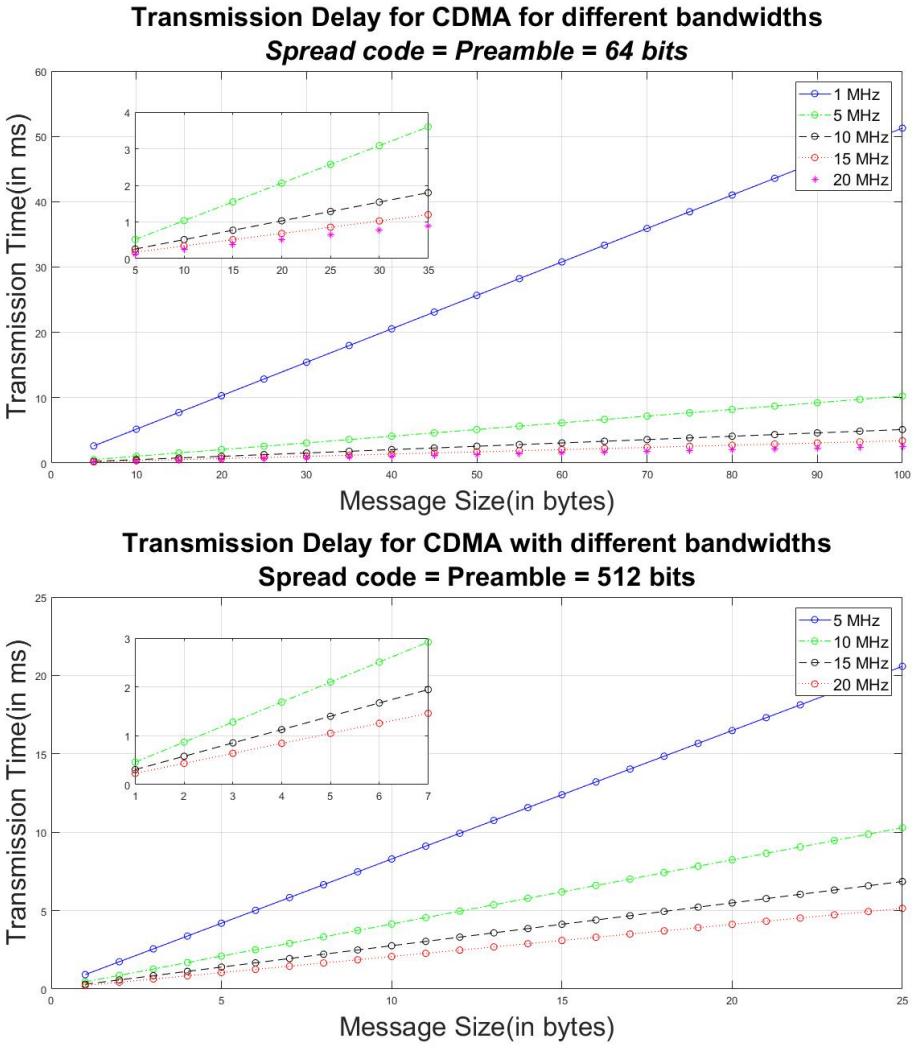
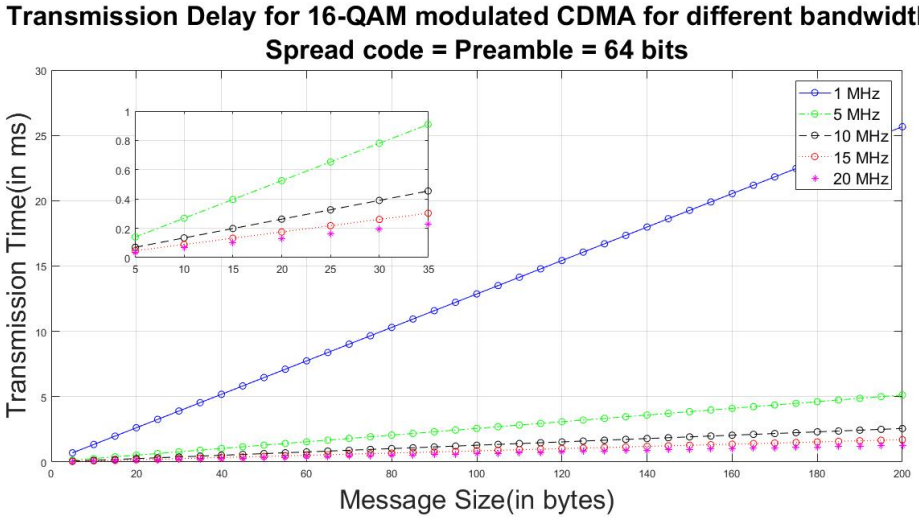


Figure 2.13: Transmission Delay (in ms) for a 16-QAM Modulated CDMA Transmission with varying spread code lengths



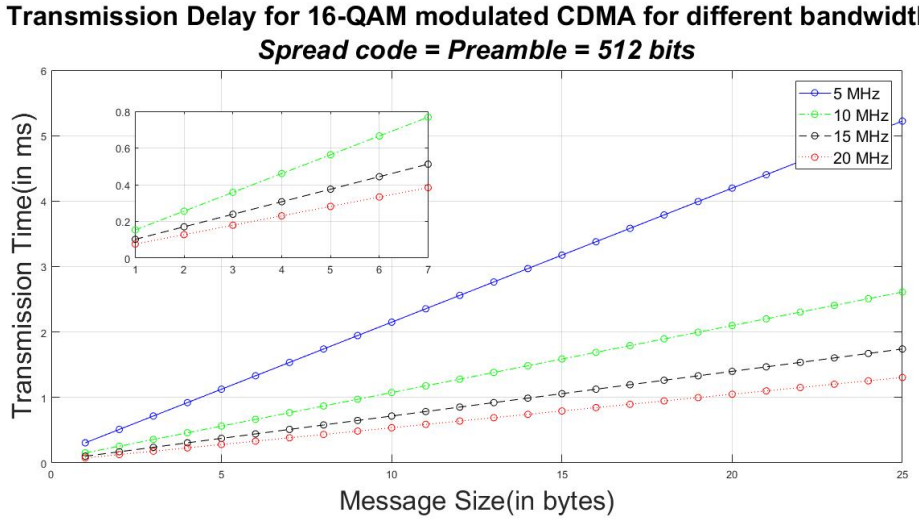


Figure 2.14: CDMA Transmission Delay (in ms) for varying spread code length

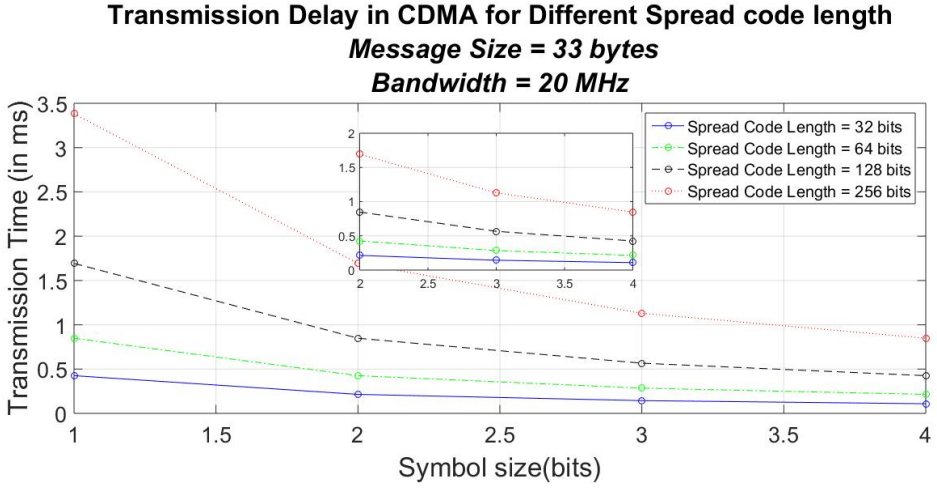
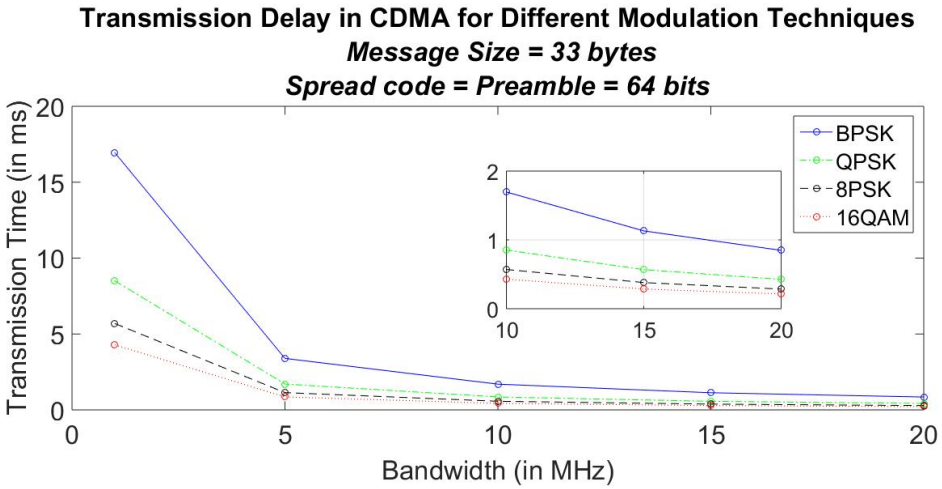


Figure 2.15: CDMA Transmission Delay (in ms) for different modulation techniques.



Parameter	Value
IoT Data Traffic Model	
IoT Application payload size (PL) distribution	Pareto distribution: $\alpha = 2.5$ (shape parameter), min $PL = 20$ bytes, max $PL = 200$ bytes (for $PL > 200$ bytes, $PL = 200$ bytes).
Periodicity split for IoT data	1 day (0.4), 2 hours (0.4), 1 hour (0.15), 30 minutes (0.05)

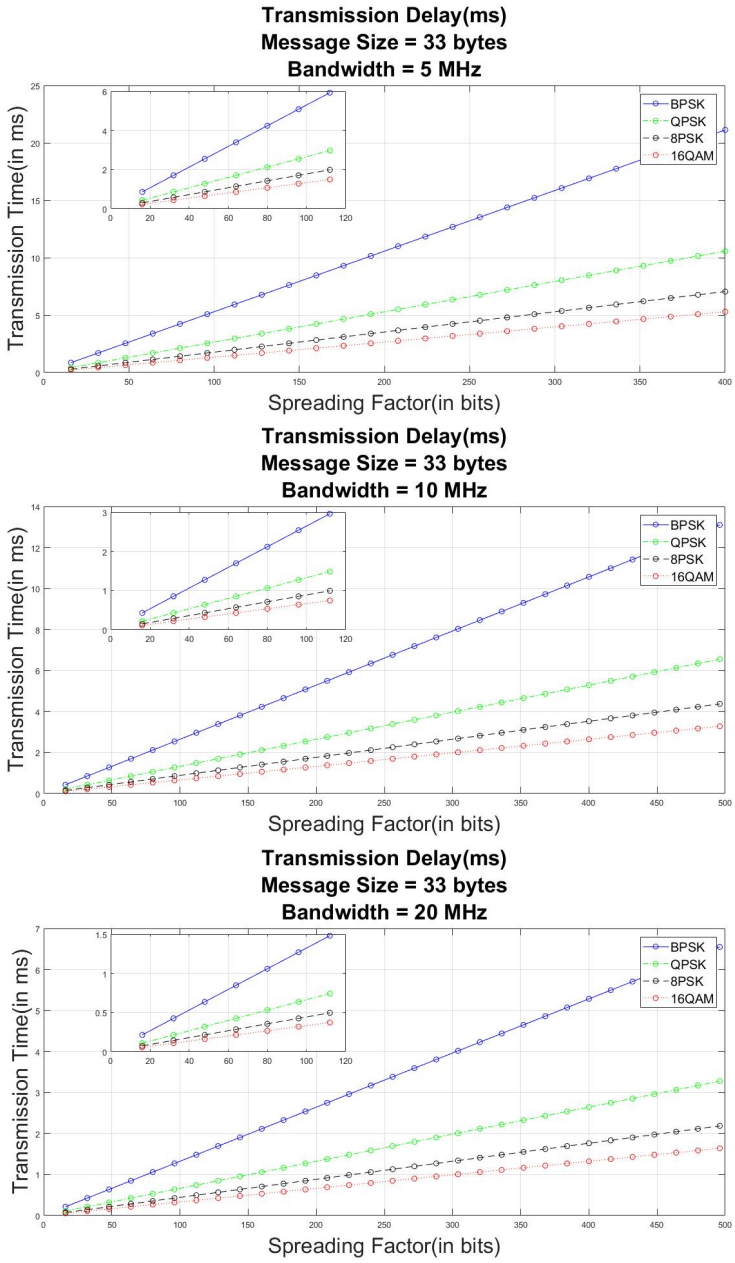
Table 2.3: IoT system parameters [2]

by calculating the mean of the Bounded(or truncated) Pareto distribution with the minimum size = 20 bytes and maximum size = 200 bytes. The formula to calculate mean with the above system parameters is :

$$Mean = \frac{L^{\alpha}}{1-\left(\frac{L}{H}\right)^{\alpha}} \cdot \left(\frac{\alpha}{\alpha-1}\right) \cdot \left(\frac{1}{L^{\alpha-1}} - \frac{1}{H^{\alpha-1}}\right) \qquad \alpha \neq 1 \tag{2.11}$$

The statistical average of the size of the IoT payload using Table 2.3 and Equation 2.11 is 33 bytes. When varying the rest of the parameters, the message size is kept constant at 33 bytes in the analysis. Transmission delay for the underlay CDMA can be seen for BPSK and 16-QAM modulated messages for varying spreading code length, bandwidth and message lengths in Figure 2.12 and 2.13 respectively. For a BPSK modulated CDMA, with spread code length of 64 and bandwidth of 20 MHz, transmission delay of as low as 5 ms can be achieved for a 200 byte message and <1ms for a 33 byte message as seen in Figure 2.15. The same values get to about 1.5 ms and 0.4ms, when

Figure 2.16: Transmission Delay (in ms) for a CDMA Transmission with varying bandwidth



modulated with 16QAM. Similarly, the delay for a constant bandwidth and varying modulation schemes, spreading code lengths is plotted in Figure 2.16. For the average message size of 33 bytes, the delays are less than 1 millisecond for a bandwidth of 20 MHz and spread code length = 64 bits as seen in Figure 2.14. As seen in Figure 2.16, for bandwidth greater than 5 MHz and spreading factors less than 256, transmission delay of <10 ms can be achieved for any modulation scheme. The point to keep in mind is that if there is an emergency packet with low latency requirement, more bandwidth and higher modulation schemes need to be provided with a BPSK or QPSK modulation scheme.

2.3.2.2 Access Delay in CDMA

Consider CDMA mobile users sharing M orthogonal traffic channels. To study the access delay in CDMA, we assume that all uplink channels are synchronized and slotted to one packet duration. Each uplink channel is assigned a unique, orthogonal spreading code such that the packets transmitted on a given channel are spread using the assigned spreading code. The analysis of slotted CDMA will be similar to the RACH procedure analysis in Section 2.3.1.2, except the message transmission. The orthogonality of preamble transmissions in RACH is analogous to the orthogonal channels in the CDMA network. The only difference is that the preambles are usually less than 64 in number, whereas the number of spreading codes assigned can be controlled to be more in number. If many UEs transmit packets in the same time slot, the result depends on two aspects [21]:

1. **Stage 1:** If all the mobiles use different spreading codes, then the receivers will be able to distinguish the packets. This will lead to successful acquisition. Whereas, users using the same spreading codes will face collision and will be retransmitted in some later time slot.

2. **Stage 2:** The successful acquisition of packets also depends on random interference introduced by the multiple transmissions associated with CDMA. Packets with zero bits in error are considered to be succesfully received and all others are

regarded as a failure. Those packets will need to be retransmitted later with some probability.

For the purpose of analysis, we assume a simplified slotted ALOHA based CDMA operation. [22, 23] The time slots are referred with the index $i \geq 0$ and are of the size of maximum packet size allowed, say P bytes. For every slot i, let X_i be a random variable which represents the number of newly arriving UEs during the slot i. Let M be the number of spreading codes available. Arriving UEs randomly select a spreading code among the M codes to spread the data packet and send it over the air. If the transmission is successful, the Base Station(BS) will send back an acknowledgement(ACK) to the user. Conversely, if the transmission fails, the UE will backoff and try to retransmit in the subsequent slot with probability p_0 using a randomly selected spreading code. The maximum number of transmissions by a UE is limited to N_{max}, as in Section 2.3.1.2.

Similar to the stochastic model for LTE in Section 2.3.1.2, we can define the random variable $M_i[n]$ for $n \in (1, N_{max})$. $M_i[n]$ represents the number of UEs engaged in their n-th transmission attempt after $(n-1)$ collisions/failures at time slot i. The process can thus be described as:

$$\begin{cases} M_{i+1}[1] = p_0.X_i \\ M_{i+1}[2] = p_0.M_{i,F}[1] \\ \ldots \quad \ldots \\ M_{i+1}[n] = p_0.M_{i,F}[n-1] \\ \ldots \quad \ldots \\ M_{i+1}[N_{max}] = p_0.M_{i,F}[N_{max}-1] \end{cases}$$

where p_0 for $0 \leq p_0 \leq 1$, is the probability of a failed UE re-transmitting in the subsequent time slot and $M_{i,F}[n]$ is a random variable representing the number of UEs that failed during their n-th transmission attempt in time slot i. Similarly, we introduce a complementary random variable $M_{i,S}[n]$ like in Section 2.3.1.2. Let us also denote $M_i = \sum_{n=1}^{N_{max}} M_i[n]$ to be the total number of UEs competing during time slot i.

The *average access delay* for the UEs with successful packet transmission, $\overline{D_{ac}}$, is

the ratio between the total access delay for all successful UEs and the total number of successful UEs, from Equation 2.7 and [4]. $\overline{D_{ac}}$ is given by

$$\overline{D_{ac}} = \frac{\sum_{i=1}^{M}\sum_{n=1}^{N_{max}} M_{i,S}[n]\overline{T_n}}{\sum_{i=1}^{M}\sum_{n=1}^{N_{max}} M_{i,S}[n]} \tag{2.12}$$

where $\overline{T_n}$ is the average delay of a UE with successful packet transmission in the n-th attempt. This includes time to transmit the first packet, the (n-1) retransmissions and the BS processing time.

$$\overline{T_n} \approx T_{Tx}(1 + (n-1)\overline{T_{Wait}}) \tag{2.13}$$

where T_{Tx} is the average transmission time of the CDMA packet as studied in Section 2.3.2.1 and $\overline{T_{Wait}}$ is the average waiting time required by a UE to perform backoff and re-transmit the packet using a different spreading code sequence. Considering that the UE fails in the first time slot, it performs random backoff and retransmits in $(1 + h)$th time slot.

$$\overline{T_{Wait}} = \sum_{h=H_{min}}^{H_{max}} q_h h \tag{2.14}$$

where, q_h is the probability of selecting the time slot $(1 + h)$ to transmit the packet after performing random backoff. H_{min} and H_{max} are the minimal and the maximal values of h, which depends on which retransmission backoff scheme was used. Some of the backoff techniques are exponential, linear, mu-law and step-function as described in detail in [24]. In comparison with the random-access delay in LTE, the access delay in CDMA seems quite less due to the following reasons:

- Reuse of spreading codes by spacing the CDMA channels in frequency domain.
- Possibility of a large number of spreading codes available.
- Absence of Message transmission and initial acquisition delay in CDMA. It is a two-message transmission.
- Flexible retransmission backoff policy.

2.4 Coexistence of CDMA Underlay and LTE Network

The underlay CDMA network is proposed to coexist with the existing LTE network in the same spectrum band. In this section, we look at an analytical model to evaluate the performance of CDMA underlay IoT and LTE network while they coexist as done in [10]. We also observe the performance of a CDMA network with a practical IoT UE deployment and data traffic model. Our focus is on the IoT uplink only.

2.4.1 Coexistence Analytical Model

Let us assume that N CDMA underlay IoT devices per CDMA channel are uniformly distributed over a hexagonal mobile site with radius r_d. Assuming IoT UEs employ power control such that eNB receive an equal power P_r^C from UEs. Assuming the worst-case IoT UE is located at distance r_d from eNB, P_t^C is the maximum transmit power and $PL^C(r_d, f)$ is the path loss at the distance r_d and frequency f, P_r^C is given as

$$P_r^C = \frac{P_t^C}{PL^C(r_d, f)}, \tag{2.15}$$

For an LTE network, we assume that M UEs are uniformly distributed over the mobile site, R resource blocks are available for LTE uplink operation and each LTE UE get M/R contiguous resource blocks. Assuming that all LTE UEs are transmitting at maximum power P_t^L, the total average power, $Q_{R_c}^L$, received at the eNB from LTE UE over CDMA channel bandwidth equivalent to R^C resource blocks is [25]

$$Q_{R_c}^L = 3\beta \frac{MR_c}{R} \frac{P_t^L}{PL^C(r_d, f)} \tag{2.16}$$

where β is the LTE channel occupancy factor and $(P_t^L M/R)$ is the transmission power per LTE resource block.

CDMA E_b/N_0 at the eNB is given by

$$E_b/N_0 = \frac{L_c P_r^C}{NP_r^C + Q_{R_c}^L + \eta} \tag{2.17}$$

and CDMA-based IoT capacity can be obtained as

$$N = 1 + \frac{L_c}{E_b/N_0} - \frac{Q_{R_c}^L + \eta}{P_r^C} \tag{2.18}$$

Parameter	Value
CDMA-based IoT Network Parameters	
Max payload size	20 Bytes
Processing gain	64 with $W = 1$ MHz
Modulation	BPSK
Information Data Rate	15.625 kbps

Table 2.4: IoT system parameters [2]

Power $(P^L_{r,W}, P^L_{r,A})$ received by eNB from the worst-case and average-case LTE UEs located at the cell boundary r_d and at average distance $r_d/2$ is given by

$$P^L_{r,W} = \frac{P^L_t}{PL^L(r_d, f)}, P^L_{r,A} = \frac{P^L_t}{PL^L(r_d/2, f)}. \tag{2.19}$$

Uplink LTE SINR (Γ^L_W, Γ^L_A) can be obtained as

$$\Gamma^L_W = \frac{K.P^L_{r,W}}{NP^C_r + \eta}, \Gamma^L_A = \frac{K.P^L_{r,A}}{NP^C_r + \eta}, \tag{2.20}$$

respectively and $K = R^C M/R$ is an overlapping bandwidth factor. LTE throughput can be computed as [26]

$$T^L_x = aW_L \log_2(1 + b\Gamma^L_x), \quad x = \{W, A\} \tag{2.21}$$

where W_L is the LTE channel bandwidth, a is a factor associated to the gap between Shannon and actual capacity and b is the LTE bandwidth efficiency.

Fig. 2.17 shows the capacity of CDMA underlay IoT network as a function of number of LTE UEs M and LTE channel occupancy β. It shows that for the worst-case, when LTE data requirement is high (e.g. $M = 100, \beta = 1$), the simultaneous CDMA underlay traffic is not possible and the IoT data traffic needs a dedicated channel assignment. For the average case, the LTE network can share the spectrum with multiple IoT UEs.

Fig. 2.18 shows the average and worst case LTE throughput as a function of number of CDMA-based IoT UEs, N. The parameters are defined in Table 2.4. As N increases the LTE throughput decreases but along with gaining the IoT UE capacity. For the average case, the LTE throughput decreases by 3% and 12% when N is 5 and 10, respectively. The analysis assumes that CDMA and LTE UEs transmit at the maximum power. But both CDMA and LTE employ power control based on CDMA/LTE co-channel interference causing lesser interference to other uplink transmission. Thus, this

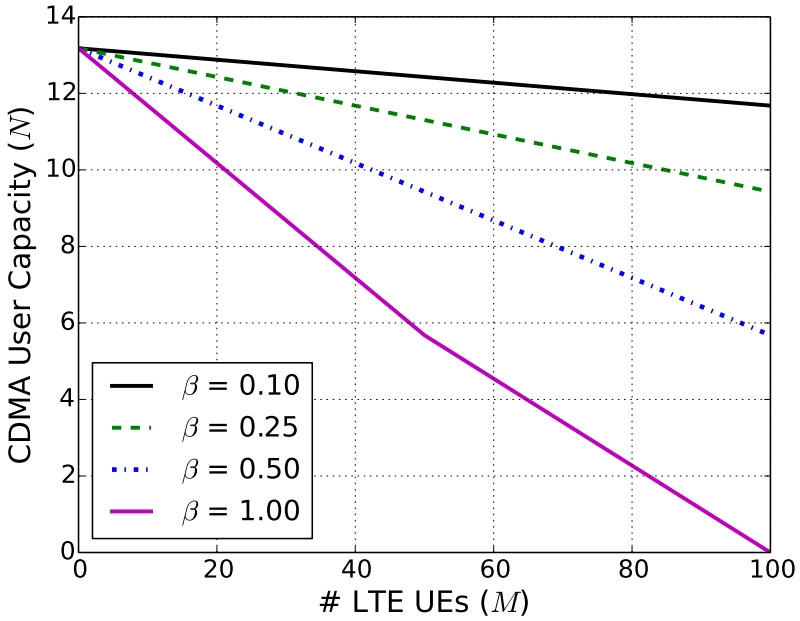


Figure 2.17: Capacity of CDMA-based IoT Network as a function of no. of LTE UEs and LTE channel occupancy (β) when required CDMA $E_b/N_0 = 7$ dB, $W = 1$MHz.

plot represents the conservative (lower bound) LTE throughput coexisting with CDMA underlay IoTs. From Figure 2.18, we see that the average LTE throughput degradation is at $\sim 3\%$ when the number of simultaneously operating IoT UEs, $N = 5$. The analysis in Section 4.2 of [2] show that this meets practical IoT data demands.

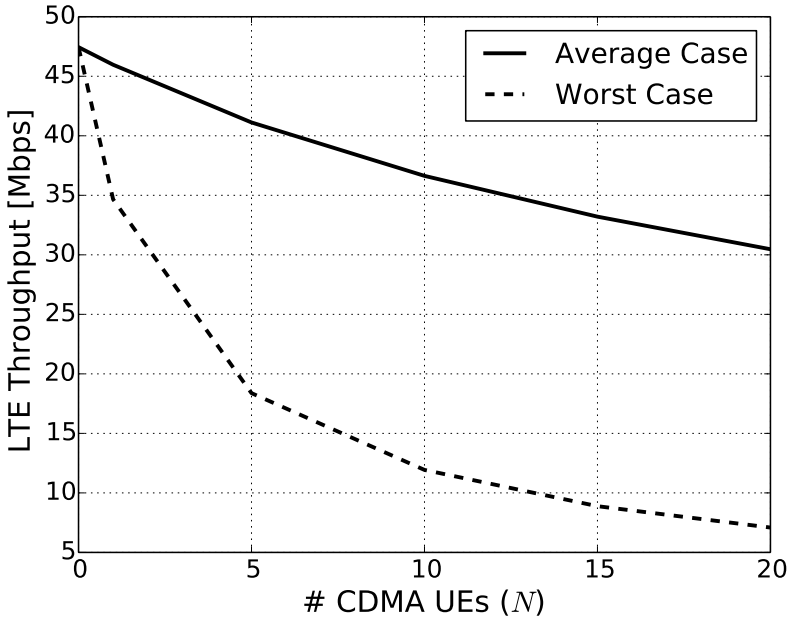


Figure 2.18: LTE throughput for a single average and worst-case user as a function of number of CDMA UEs

Chapter 3

System Design

3.1 System Requirements for IoT at PHY/MAC layers

The main objective at MAC layer is to schedule IoT traffic with the minimum wait time to access the channel. At physical layer, we needed to identify an optimal waveform operating at low signal-to-interference-plus-noise ratio (SINR) and bit-error-rate (BER) and without causing significant interference to the overlay LTE transmission. Considering these requirements, the CDMA underlay transmission becomes a favorable choice for IoT services, as shown in Figure 3.1 as:

- It does not require any UE-eNB handshaking (RACH+RRC) contrary to earlier efforts made in the community.
- Asynchronous CDMA transmission enables decentralized spectrum access at IoT UEs which reduces wait time to get assigned resources from eNB.
- CDMA IoT transmission can reject narrow band OFDMA LTE interference without causing significant interference to LTE as well with its low-power transmission.
- CDMA-based IoT transmission utilizes the legacy CDMA support available at cellular network.

3.2 CDMA Design

The design of the CDMA(IoT) Transmitter and Receiver is explained in this section. Fig. 3.2 shows a packet format instance for the uplink IoT data transmission. The CDMA MAC frame consists of source address (or ID), message length, payload and

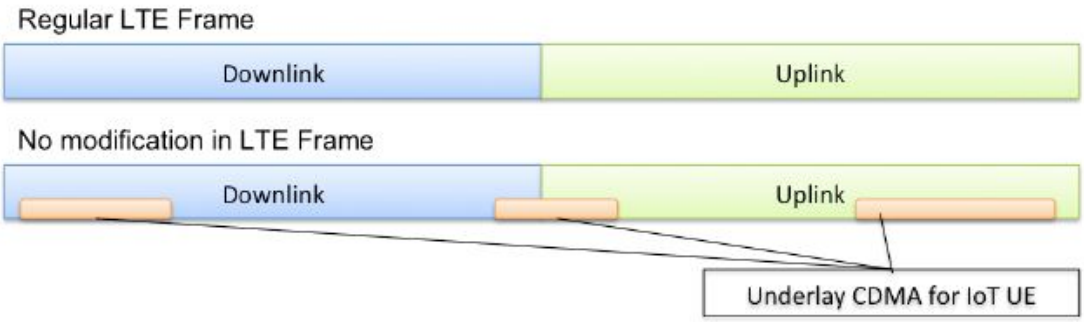


Figure 3.1: Underlay CDMA Transmission for IoT-Uplink

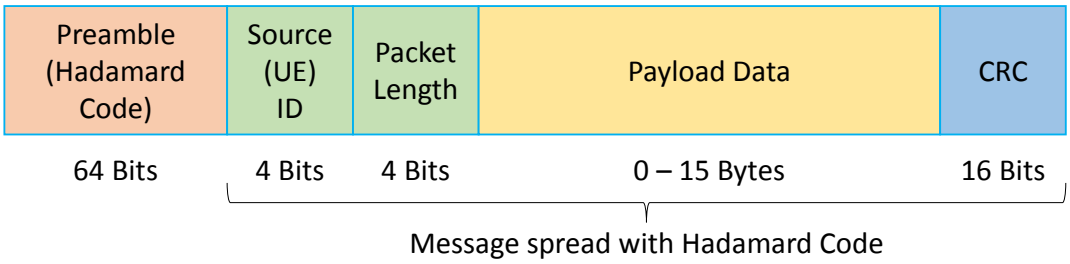


Figure 3.2: Packet format of CDMA-based IoT data transmission

cyclic Redundancy Check (CRC) to check errors in frame. This example frame format allows to send variable size payload of 0-15 Bytes which is typical IoT traffic profile emanating from sensing devices. The MAC frame is spread/despread with a Walsh-Hadamard code (of length 64 bits in this case) where different codes are assigned to different UEs. Data is modulated/demodulated using BPSK to be able to operate on low-SINR. Packet preamble contains a unique Walsh-Hadamard code which is used to detect the start of the packet at the receiver by correlating the signal with the known code. Zadoff-Chu sequences were avoided in the design to avoid interference with the LTE synchronization signals, which use ZC sequences. The CDMA Transmitter blocks are shown in Figure 3.3

At the Receiver, as shown in Figure 3.4, **cross-correlation** of the received samples is done with the expected Preamble code.

The cross-correlation is a sliding product of two signals. It is most commonly used to measure the similarity between the two input signals. The cross-correlation, R_{xy} of

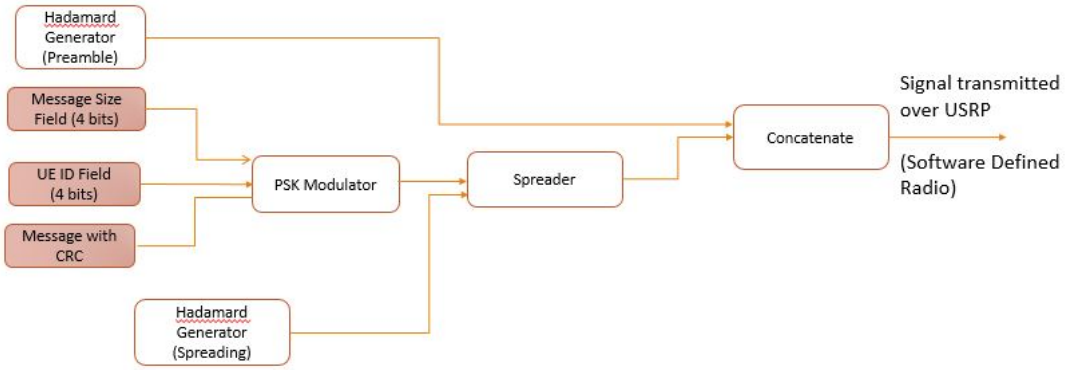


Figure 3.3: CDMA Transmitter Block Diagram

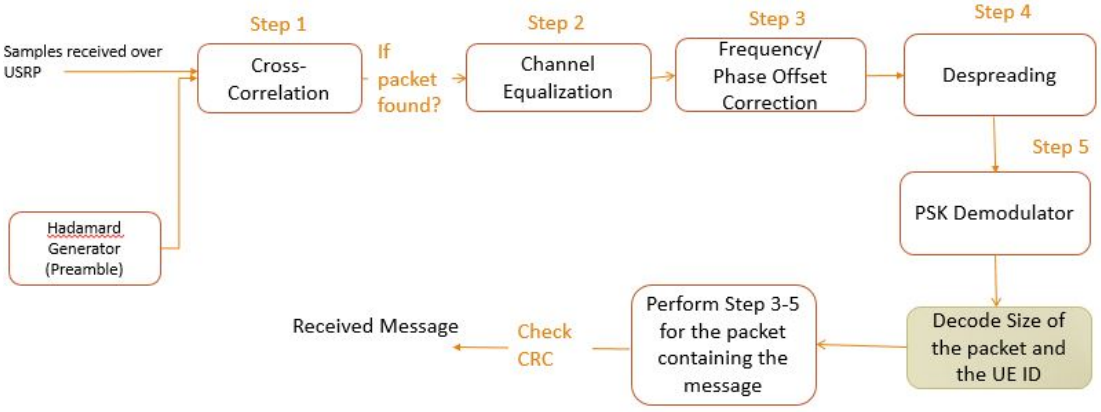


Figure 3.4: CDMA Receiver Block Diagram

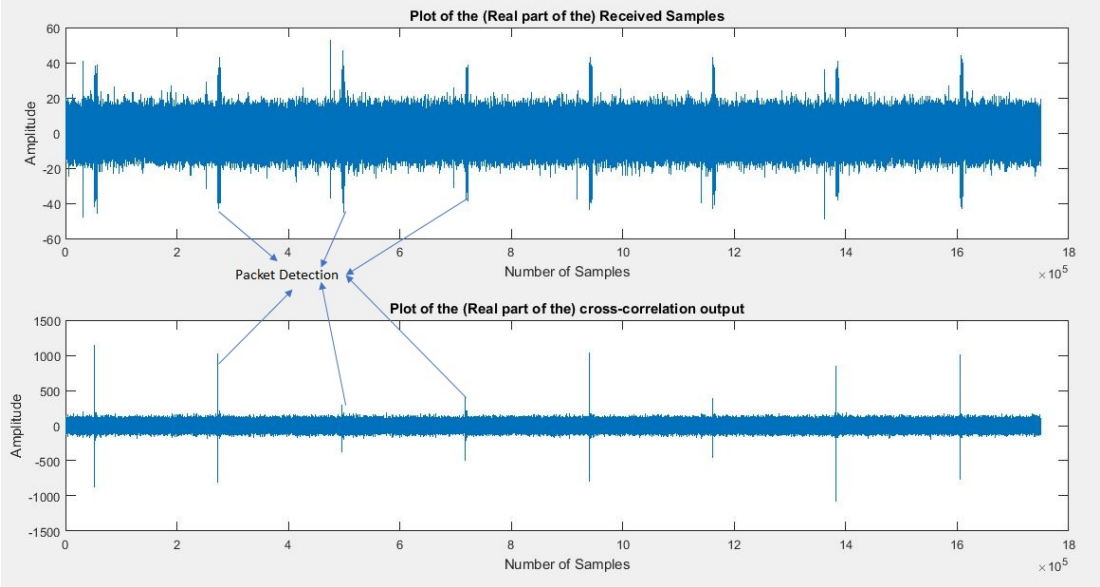


Figure 3.5: Correlation output of the Received Samples

two jointly stationary random processes x_n and y_n is given by :

$$R_{xy}(m) = E[x_n y^*_{n-m}] = E[x_{n+m} y^*_n] \tag{3.1}$$

where $\infty < n < \infty$, the asterisk denotes complex conjugation, and E is the expected value operator. [refer mathworks source for xcorr]

If the output of the correlation, as can be seen in Figure 3.5 is greater than a threshold(noise), that signifies packet detection. This threshold is set by the user and should be higher than the noise floor. Currently, it is set using a trial-and-error method but can be made dynamic in future. Once the packet is detected, **channel equalization** is done for the UE ID and the sizeField length (512 bits for a 64-bit spreading sequence). It compensates for the Inter-symbol Interference (ISI) caused by multipath using a training sequence, in this case the preamble. We do not implement the RAKE receiver in this case, and thus have to equalize the channel to decode the packets correctly.

A linear adaptive filter, called Recursive Least Squares(RLS) is used to do the channel estimation and equalization. This filter finds the coefficients recursively that minimize a weighted linear least squares function relating to the input signals, as can be seen in Fig. 3.6. RLS exhibits extremely fast convergence at a cost of high computation complexity.

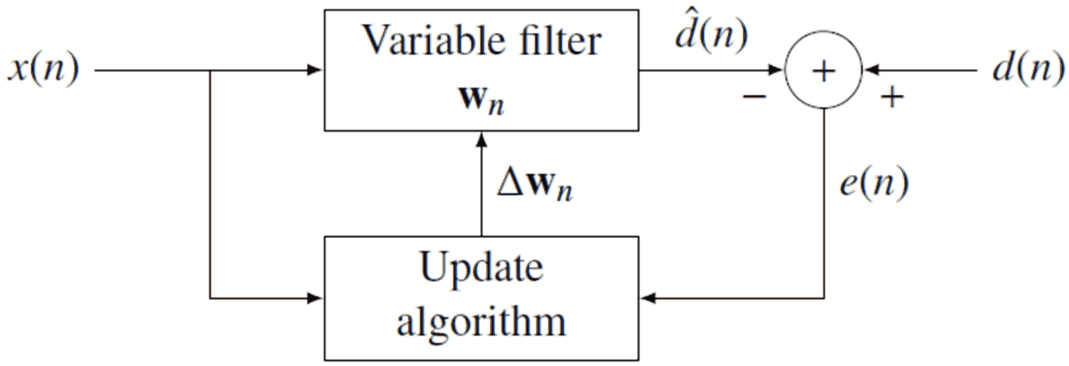


Figure 3.6: Recursive Least Square filter

Δw_n is the filter coefficient, $e(n)$ is the error signal and $d(n)$ is the reference signal. The update algorithm for a p-th order RLS filter is summarized as: $Parameters$:

- p = filter order
- λ = forgetting factor
- δ = value to initialize $P(0)$

$Initialization$:

- $w(n) = 0$
- $x(k) = 0$, $k = -p, ..., -1$
- $d(k) = 0$, $k = -p, ..., -1$
- $P(0) = \delta^{-1}I$ where I is the identity matrix of rank $p + 1$.

$Computation : For n = 1, 2, ...$

$$\mathbf{x}(n) = \begin{bmatrix} x(n) \\ x(n-1) \\ \vdots \\ x(n-p) \end{bmatrix}$$

$$\begin{aligned}
\alpha(n) &= d(n) - \mathbf{x}^T(n)\mathbf{w}(n-1) \\
\mathbf{g}(n) &= \mathbf{P}(n-1)\mathbf{x}(n)\left\{\lambda + \mathbf{x}^T(n)\mathbf{P}(n-1)\mathbf{x}(n)\right\}^{-1} \\
\mathbf{P}(n) &= \lambda^{-1}\mathbf{P}(n-1) - \mathbf{g}(n)\mathbf{x}^T(n)\lambda^{-1}\mathbf{P}(n-1) \\
\mathbf{w}(n) &= \mathbf{w}(n-1) + \alpha(n)\mathbf{g}(n)
\end{aligned} \tag{3.2}$$

The equalization is followed by **Carrier Frequency Offset Correction**. Carrier frequency offset often occurs when the local oscillator signal for down-conversion in the receiver does not synchronize with the carrier signal contained in the received signal. This leads to the received signal being shifted in frequency. It is a two step carrier synchronization, coarse and fine. Coarse frequency offset estimation is done by the *PSKCoarseFrequencyEstimator* object in MATLAB. The algorithm used to estimate the coarse frequency offset is [27]. The estimate frequency and phase offset is then applied to the input signal using the $PhaseFrequencyOffset$ object. The fine frequency offset correction is done by $CarrierSynchronizer$ object, which is a closed-loop compensator that uses a PLL based algorithm described in [28] can be seen in Figure 3.7. The output of the synchronizer. y_n is a frequency shifted version of the complex input signal, x_n. The synchronizer output is :

$$y_n = x_n e^{i\lambda_n} \tag{3.3}$$

where λ_n is the output of the direct digital synthesizer, DDS and the e_n is the phase error for the n^{th} symbol. The DDS is the discrete-time version of a voltage-controlled oscillator. In this context, the DDS acts as an integration filter. The error signal, e_n for a BPSK signal is given by:

$$e_n = sgn(Re\{x_n\}) * Im\{x_n\} \tag{3.4}$$

Next step is to de-spread the equalized and offset corrected signal using the Hadamard sequence that was used to spread it. In our design, we pass the signal through despreading operations for different Hadamard spread codes. For simplicity, the Hadamard code index is chosen to be the same as the UE ID. After the despreading, if the UE ID is the same as the code index, the signal is then demodulated to get the size of the message

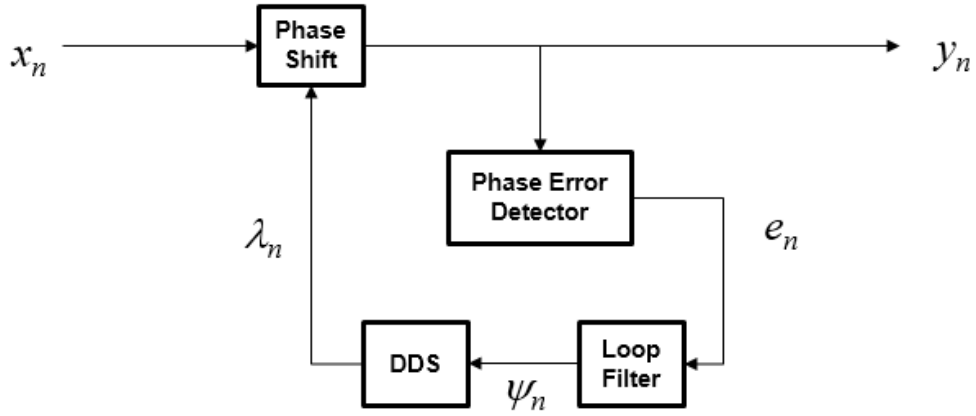


Figure 3.7: Carrier Synchronizer Block Diagram

received and the ID of the UE (Transmitter). If the UE ID retrieved is not the same as the Hadamard code index, the signal is despread with the next Hadamard code index.

Once the size of the message is retrieved and the transmitter UE's ID is known, we get to know the client UE which is transmitting the packet. The rest of the packet is decoded using the known despreading code sequence and the same steps are followed as explained above to decode the final bits.

Cyclic Redundancy Check (CRC) is an error detecting code which is used to detect possible errors in bits received and determine if the message is received in error. The transmitter calculates the checksum, which is a function of the input message and is appended to the message bits. The receiver uses the same function to calculate the CRC checksum bits and is then the checksum is compared with the received checksum. A generator polynomial is used to generate the CRC bits and we use the CRC-16 polynomial given by $x^{16} + x^{15} + x^2 + 1$ or $0x8005$.

The CDMA Packet in time domain is represented in Figure 3.8. The CDMA channel uses 1 MHz bandwidth with data rate of ∼10 - 15 Kbps for a spreading code length of 64 bits. The proposed design parameters are consistent with existing low power wide area network IoT technologies and also compatible with coexistence operation of LTE. In Figure 3.9 is a representation of the CDMA packets as underlay transmissions with a Signal-to-Interference Ratio of -3 dB.

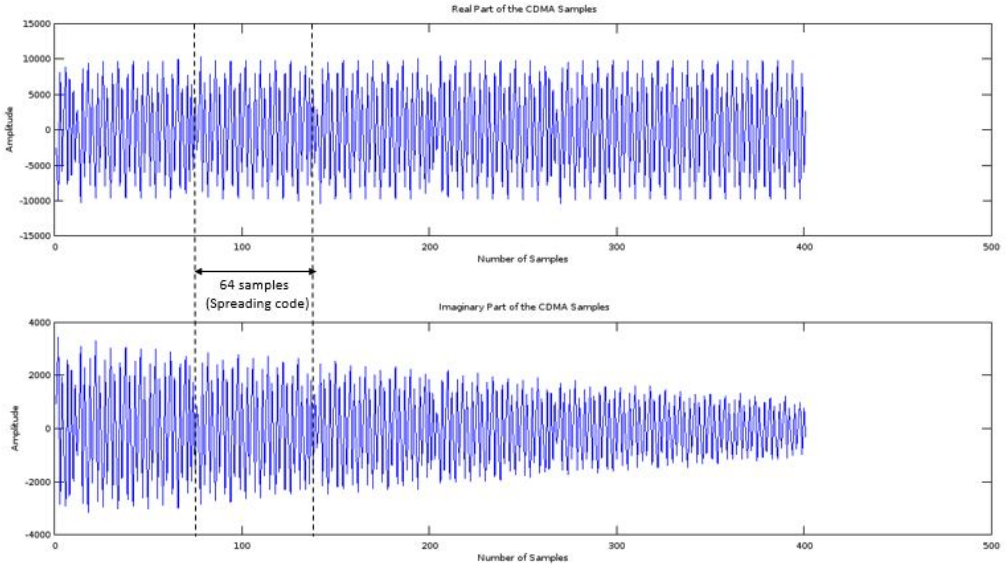


Figure 3.8: Time Domain Representation of the CDMA Packet at the Receiver

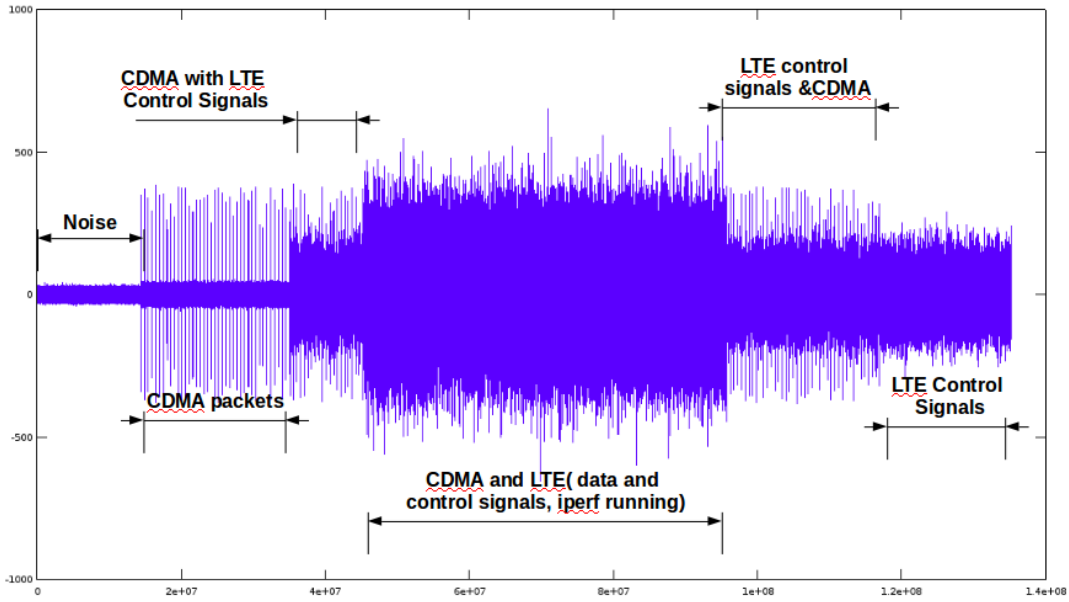


Figure 3.9: Time Domain Representation of the CDMA co-existing with LTE

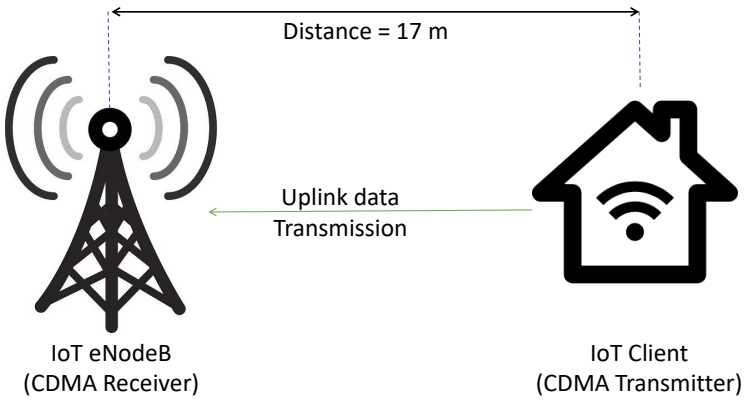
Chapter 4

Implementation and evaluation

4.1 CDMA Implementation

CDMA-based IoT transmission is prototyped using a software-defined radio (SDR) platform using GNU radio and Universal Software Radio Peripheral (USRP). USRP Hardware Drivers (UHD) are used to transmit/receive samples to/from the USRP. The CDMA transmitter and receiver code is developed on top of UHD in C++ and C, respectively, to process and decode CDMA packets in real time. Intel(R) Core i7 4th Generation CPU (@3.60 GHz) machines are used in performance mode. Developing the CDMA receiver is particularly challenging to detect CDMA packets considering random wireless channel and critical time constraints of real-time signal processing. For example, to detect beginning of the packet, we perform cross-correlation of a known 64-bit preamble and 10,000 received samples (equivalent to size of the maximum size of the packet, i.e. 9280 samples) at one instance. Each instance takes processing time of 160 ms including 100 ms for reading samples at sampling rate of 1 MSps and 60 ms for running correlation function. We choose sampling rate 1 MSps to avoid overflowing of samples at receiver which is caused due to higher sampling rates. This parameter also restrict the maximum achievable data rate for the CDMA transmission. Furthermore, a packet is detected if the peak value of the cross-correlation output is greater than certain threshold which is a function of SINR. Here the choice of the threshold becomes critical. If it very low, then there is significant false packets detection which eventually get discarded while checking the packet CRC. At higher threshold values, packets gets missed in the detection function. This threshold is controlled manually by the user, and is set to a value a little above the noise floor.

We evaluate our prototype for the following scenarios:



chch

Figure 4.1: Setup of a single IoT link for uplink transmission using CDMA software implementation and USRP

(1) standalone operation of CDMA IoT transmission and, (2) coexistence of CDMA IoT and LTE transmissions. IoT nodes are realized using USRP series N210 and/or B210. LTE transmission is enabled using OpenAirInterface (OAI) where OAI is a PC-hosted open sourced SDR platform [29]. The LTE UE is connected to the eNB using FDD mode, 5 MHz bandwidth and transmission mode 1 (SISO).

4.1.1 Standalone operation of CDMA IoT

We have chosen two nodes in ORBIT grid as the IoT eNodeB and IoT Client, where the distance between the two chosen nodes was 17m. The setup can be seen in Figure 4.1.

Figure 4.2 shows the packet error rate (PER) of a single uplink CDMA IoT transmission as a function of Signal-to-Noise-Ratio (SNR). In our experiment, we varied packet payload size, PL, as $\{10, 12, 15\}$ Bytes. For each PL, packets are transmitted at the interval of 60 milliseconds and, with the constant receiver gain, transmitter gain is adjusted to vary SNR between 0 to 8 dB. We observed that PER is between 0.22 to 0 for SNR range from 0 to 4.5 dB and PER is always zero for SNR above 4.5 dB. Also, PER does not get affected significantly by the PL values considered in the experiment. Through these set of experiments, we show that our prototyped CDMA IoT transmission can transmit data with low PER with significantly low SNR value along with varying packet payload size.

In the next set of experiments, we vary the size of the Hadamard code that is used

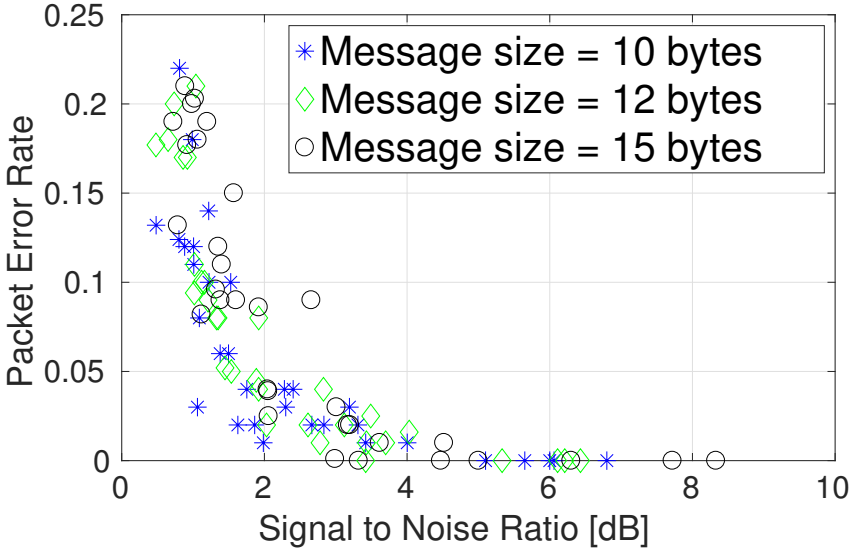


Figure 4.2: Packet Error Rate of a CDMA-based IoT transmission as a function of the Signal-to-Noise Ratio with varying message sizes

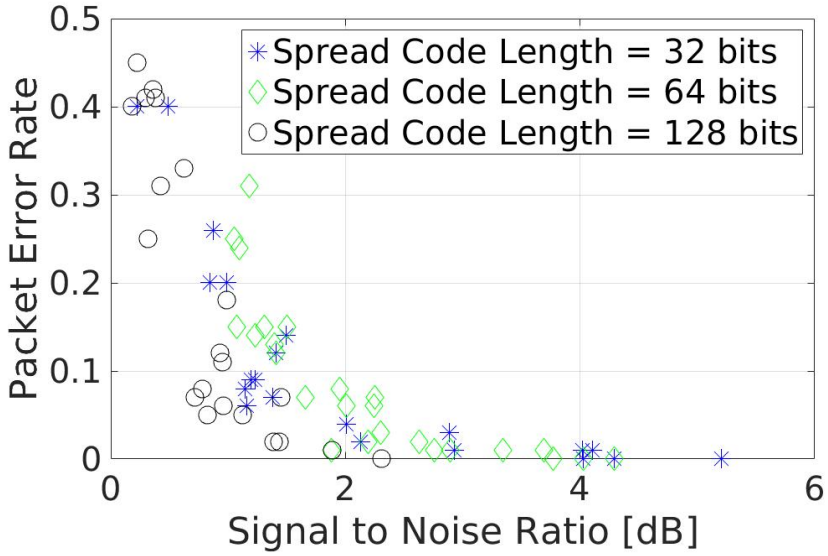


Figure 4.3: Packet Error Rate of a CDMA-based IoT transmission as a function of the Signal-to-Noise Ratio with varying Spreading Code length.

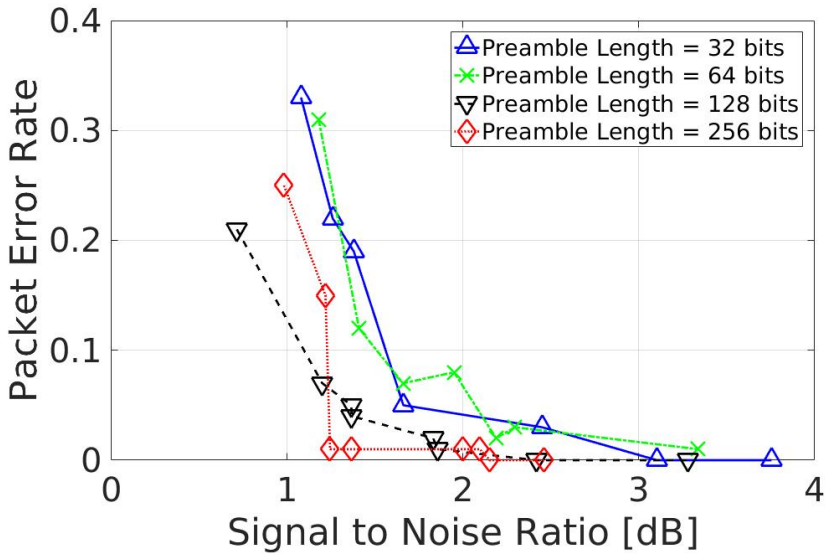


Figure 4.4: Packet Error Rate of a CDMA-based IoT transmission as a function of the Signal-to-Noise Ratio with varying Preamble length.

to spread the input message. With an increase in spreading code length, the packet is spread over a wider bandwidth. The spreading code length, S_l is varied in $\{32, 64, 128\}$ bits. For each S_l, packets are transmitted at an interval of 50 milliseconds. Similar to the above experiment, we change the transmit gain to vary the SNR between 0 to 5.21 dB. We observe, as can be seen in Figure 4.3 that there is no significant improvement in performance for $S_l = 32$ and 64 and both achieve zero packet error rate (PER) at $\sim$ 3.8 - 4.2 dB. For $S_l = 128$, there is an evident increase in performance, as zero packet error rate (PER) is attained at 2.31 dB. From the results of these experiments, we show that the CDMA transmission can transmit with a lower PER at a particular SNR, as the spreading code length increases.

The next set of experiments involve varying the preamble length in the packet. Preamble is a Hadamard code sequence, which is required to find the start of the packet at the receiver. The preamble length, Pr_l is varied in $\{32, 64, 128, 256\}$ bits, which can be seen in Figure 4.4. The SNR is varied from 0 to 3.8 dB and the Packet Error Rate (PER) of the packets is studied for different lengths. For preamble lengths of 32 and 64, the PER remains in a close range for a specified SNR and has no remarkable improvement. Whereas, as we increase the preamble length to 128 bits and 256 bits,

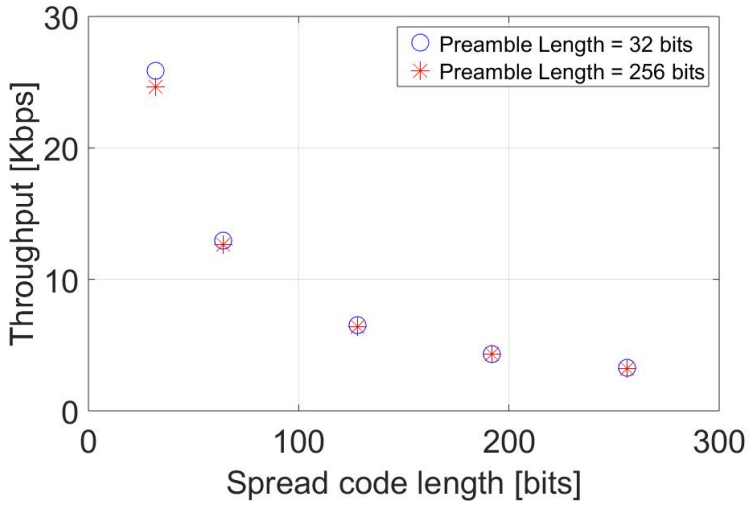


Figure 4.5: CDMA Throughput for varying spreading code lengths and keeping bandwidth constant at 1 MHz

we observe the PER to go down for low SNR values and exhibits better performance.

Now, we look at the CDMA throughput with varying spreading code length for a bandwidth of 1 MHz. The length of the preamble does not affect the throughput appreciably since it is a very small part of the CDMA packet transmitted. In Figure 4.5, the maximum message size is 15 bytes(or characters). Maximum achievable throughput is 25.86 Kbps, with spread code length = 32 bits. High throughput comes at the expense of higher Packet Error Rate (PER) at low SNRs. We can increase the maximum size of the message sent to 255 bytes, augmenting the throughput to upto 30 kbps. It can also be understood from the Figure 4.5 that for different preamble sizes, the throughput largely remains unaffected. CDMA throughput for varying bandwidth is plotted in Figure 4.6. The CDMA system is capable of providing a maximum throughput of about 258.6 Kbps if it occupies 20 MHz bandwidth and the spreading code length is 64. Throughput efficiencies for spread code lengths of 64 and 512 are 0.01293 bps/Hz and 0.001626 bps/Hz, respectively.

We now look at T_{total}, the total delay for transmission of a CDMA packet by an IoT UE.

$$T_{total} = T_{access} + T_{transmit} \tag{4.1}$$

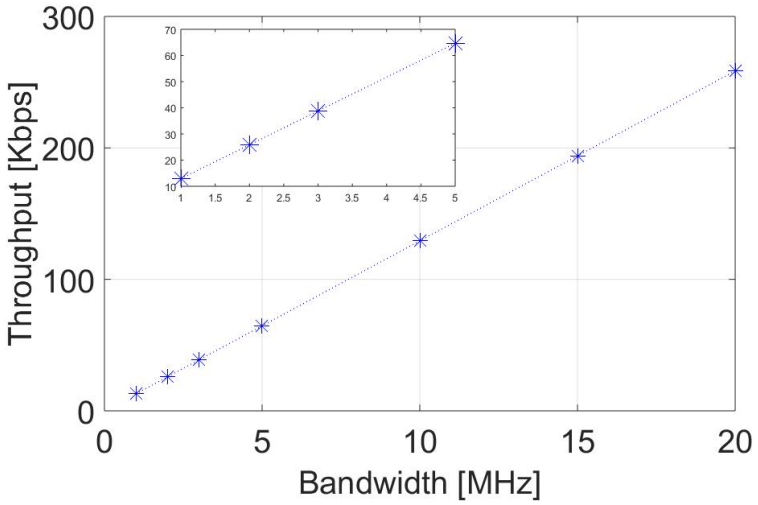


Figure 4.6: CDMA Throughput for varying bandwidths keeping spreading code length constant to 64 bits

where, T_{access} is the time required to acquire the preamble and confirm packet detection. $T_{transmit}$ is the delay in successfully transmitting the packet, which includes $T_{retransmission}$, delay added due to error in packet detection at the receiver and $T_{transmission}$, the time to transmit the CDMA packet once. $T_{retransmission}$ depends on the packet error rate, when operating at a certain SNR and $T_{backoff}$, backoff time. $T_{backoff}$ is ignored in out analysis.

Consider a BPSK modulated CDMA transmission, with preamble and spread code length to be 64 bits operating at 1 MHz bandwidth. From Figure 4.3, we get the packet error rate(PER) of 0.15 at SNR of 1.5 dB. $T_{preamble}$ is the time to acquire 64 bits in 1 MHz bandwidth = 64 microseconds. N is the average number of retransmissions made by a UE and is equal to $\frac{1}{1-PER}$.

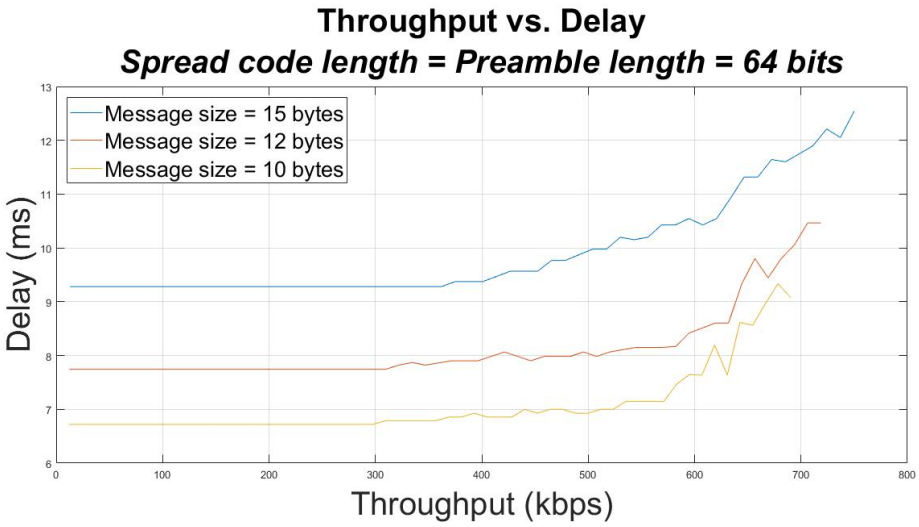
$$T_{Transmit} = \frac{T_{transmission}}{1 - PER} + T_{backoff} \approx \frac{T_{transmission}}{1 - PER} \tag{4.2}$$

$T_{transmit}$ gives the delay experienced by users transmitting CDMA packets. These users need to have power control and simultaneous packet transmissions need to be managed to restrict the bit-error rate as per Equations 2.4 and 2.5.

The throughput of the system, T is given by:

$$T = R * N; \tag{4.3}$$

Figure 4.7: Throughput Vs Delay plot for different message sizes (experimental)



where, R is the data rate per CDMA signal and N is the number of UEs transmitting simultaneously. For a range of N, the SINR is calculated from equation 2.4.

Next step is to plot some Throughput vs Latency(Delay) graphs for various CDMA message sizes, preamble lengths and spreading code lengths. The packet error rates(PER) required in calculating the delays is fetched from the experimental results given above. These plots will help us in figuring out the practical value of achievable throughput with considerably minimal delay. These values are a little optimistic, considering the experiments only had one user transmitting at low SINR. When there are a large number of UEs transmitting, the interference from other users can be a little difficult to predict. These plots are done for preamble length and spread code length at 64, if not mentioned otherwise. The CDMA bandwidth is 1 MHz.

In Figure 4.7, the delays remain constant at 9.28 milliseconds for the initial range of throughput. This is the range where the SNR is high and the PER is 0. Then, there is a gradual increase in the delay as the throughput of the system increases. For message size = 15 bytes, there is a considerably steep rise in the delay between 500-700 kbps, where the delay rises from 10 ms to 12 ms. As mentioned before, these numbers are a bit optimistic considering there is only one transmission. Similarly, for message sizes 10 and 12 bytes, the delay has a significant rise in the 600-700 kbps throughput range. This range of throughput is achieved for number of UEs in the range 40 to 50.

Figure 4.8: Throughput Vs Delay plot for different message sizes (analytical)

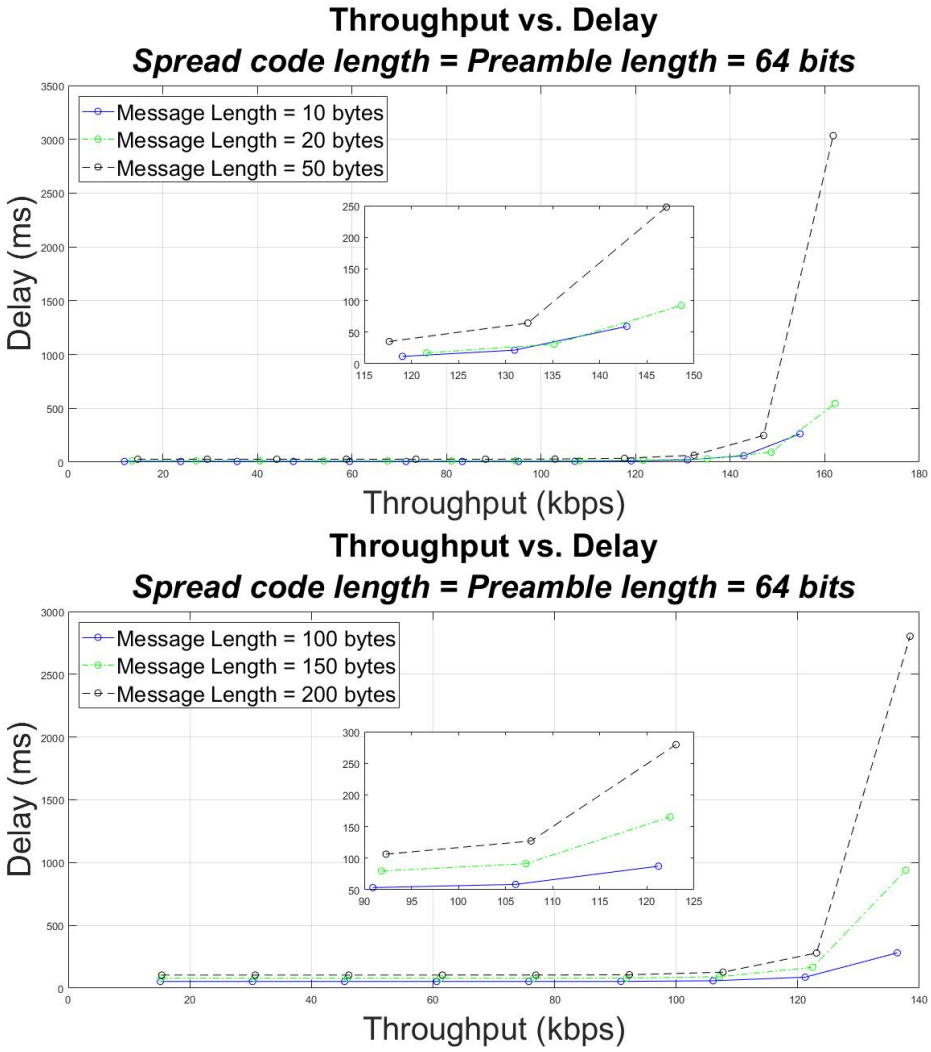
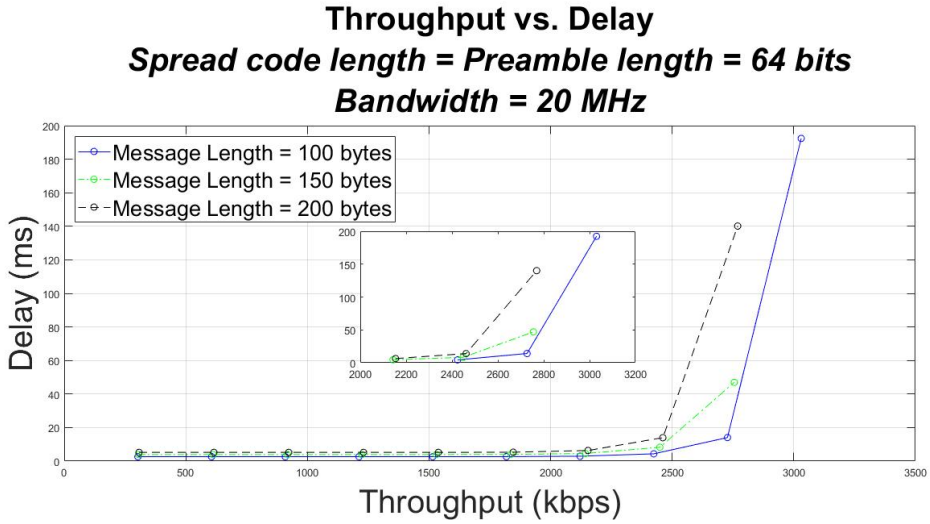


Figure 4.9: Throughput Vs Delay plot for different message sizes and higher bandwidth



The plot is also done analytically in Figure 4.8. In the first subfigure with message sizes ≤ 50, the rise in delay is steep for throughput range 120 - 160 kbps. The corresponding N values for a 10 byte message are 10 to 14. The delay at throughput ~ 120 kbps is about 12 ms for a 10 byte message. For the 50 byte message, the delay is 63 ms for throughput 130 kbps. These values are pretty low compared to the total acquisition and connection establishment latency for all UEs combined. In the second sub-figure with messages sizes ≥ 100, the steep rise in delay is for throughput 105 - 140 kbps. For the 200 byte message, the delay is 127 ms and 279 ms for throughputs 107 kbps and 123 kbps, respectively. Thus, the number of simultaneous UEs need to be limited to 7 for delays less than 100 ms and throughput of $\sim$100 kbps. If CDMA is provided with more bandwidth and the message size is 200 bytes, throughput of 2460 kbps can be achieved with the latency of 14 ms as shown in Figure 4.9.

Next set of Throughput Vs Latency graphs are for variable preamble length and spreading code length. Delays for different preamble lengths in a CDMA packet is shown in Figure 4.10. The delay rises considerable between 600 to 700 kbps, increasing from 9.5 ms to about 13-14 ms in that range. Since the preamble is a very small part of the whole packet, the delays behave similarly for different preamble lengths as seen in the second subfigure of Figure 4.10. The throughput with considerably low delay is

Figure 4.10: Throughput Vs Delay plot for different preamble lengths (experimental and analytical)

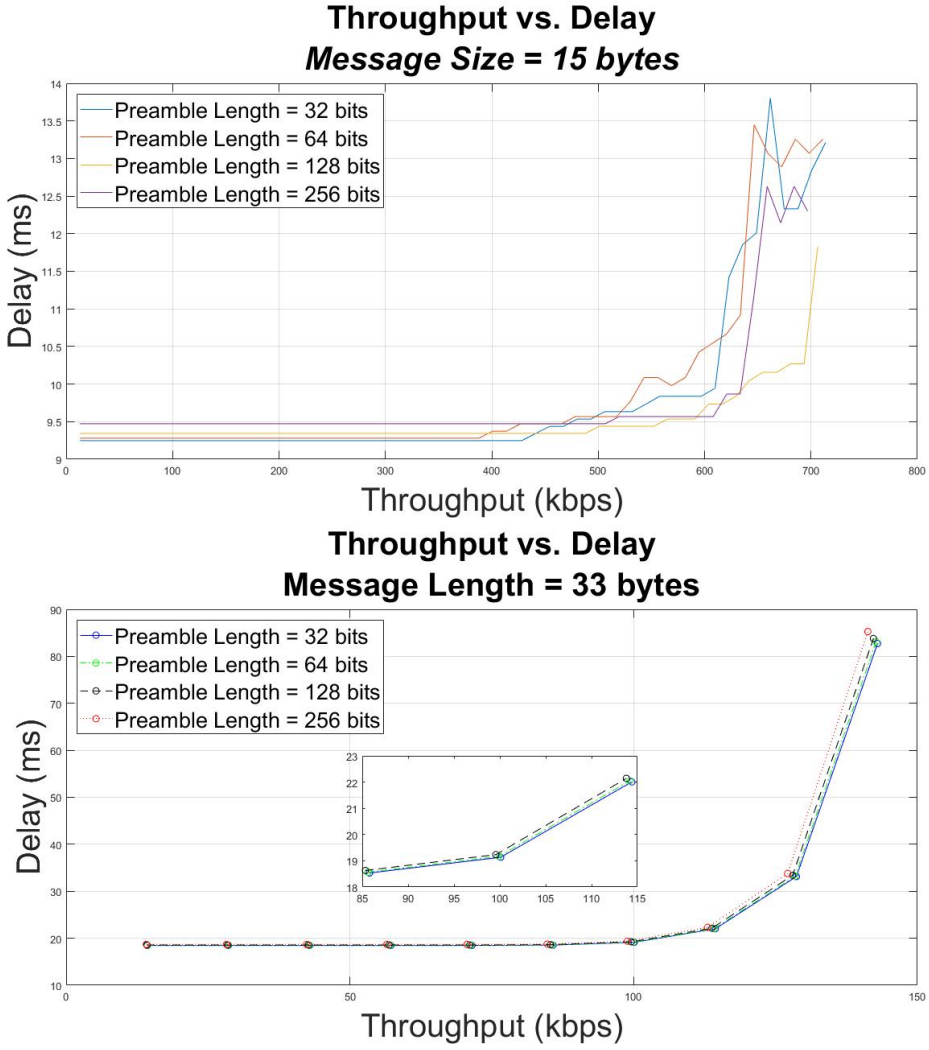
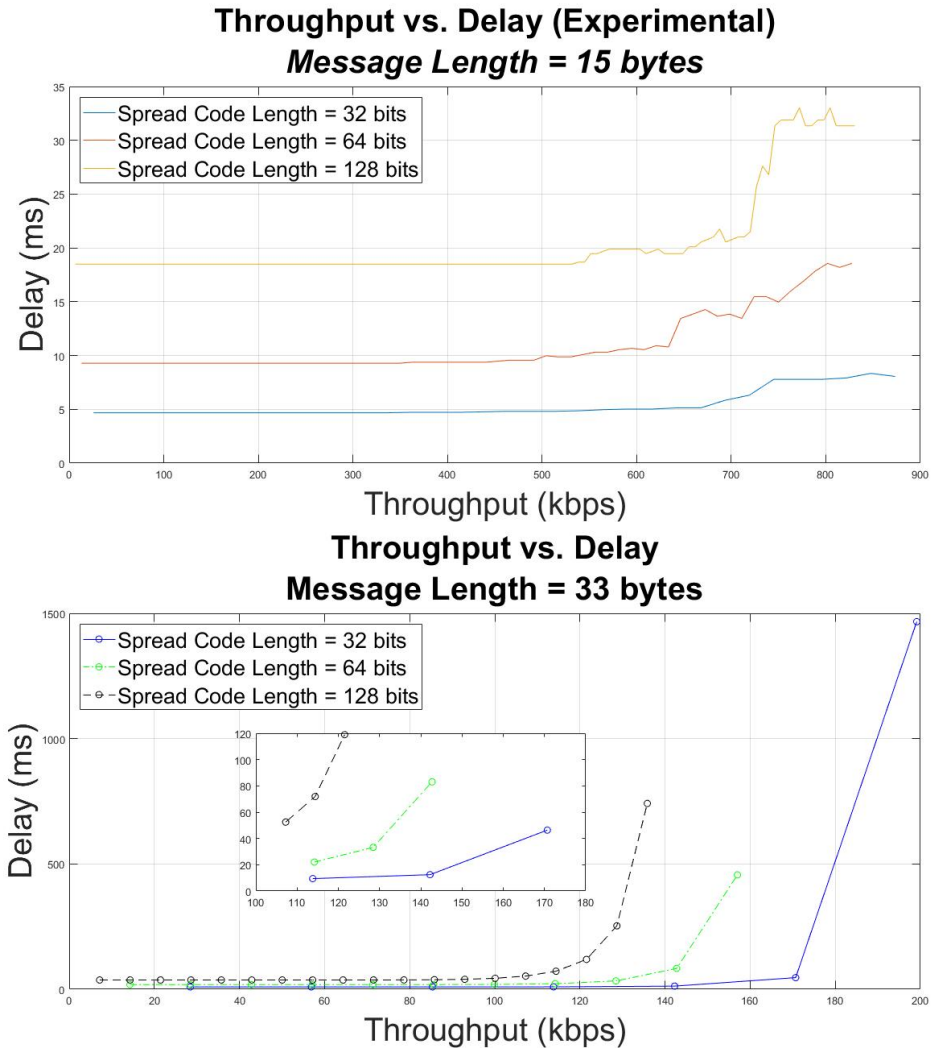


Figure 4.11: Throughput Vs Delay plot for different spreading code lengths (experimental and analytical)



between 85 and 130 kbps, for which delays are between 19 ms and 33 ms.

Figure 4.11 shows throughput vs. delay plots for varying spreading code lengths. Using lower spreading code lengths will have much lower transmission delay at the same throughput value. This is because lower spread code lengths will operate at higher SINR values, and thus will support less number of UEs. Based on our experiments, the correct range of throughput values are between 600-700 kbps in the first subfigure, for low transmission delay ($< 20ms$) and to support acceptable number of users. In the second subfigure of Figure 4.11, we can see that with the spreading code length of 128 bits, large number of users can be supported at the cost of little high delay. Depending

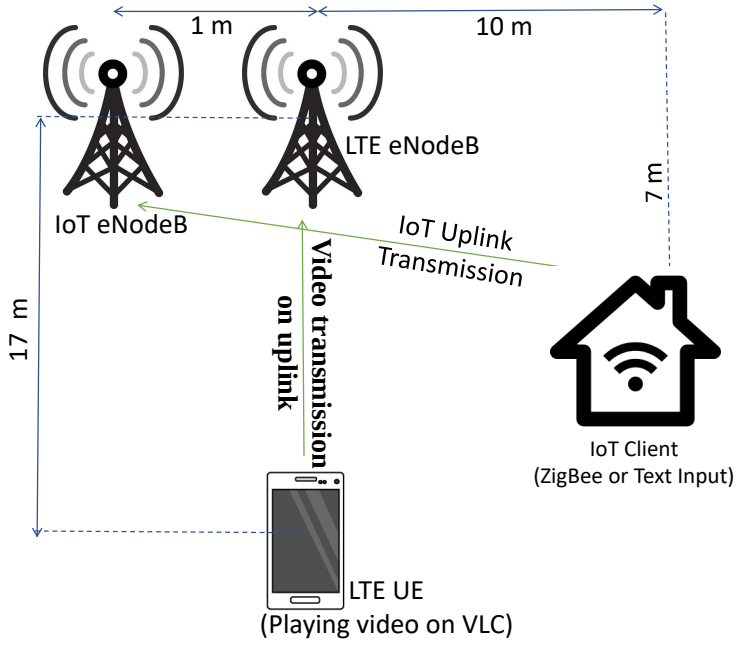


Figure 4.12: Setup for coexistence of IoT link and LTE using CDMA-underlay implementation, OAI and USRP

on the delay requirements, the spreading code lengths can be controlled. For reliability and high occupancy applications, length of 128 bits should be considered. Whereas for low latency applications, higher bandwidth and a 64 bit spreading code could be suitable.

4.1.2 Coexistence of underlay CDMA with LTE

OpenAirInterface(OAI) is used to implement the current LTE system on a USRP, with OFDM based downlink and SC-FDMA based uplink frames for regular LTE clients. The LTE UE is connected to the LTE eNB using FDD mode, 5 MHz bandwidth, transmission mode 1 (SISO), and MCS index of 9 for uplink / downlink (QPSK modulation). At the same time, the IoT clients, as developed in previous phase, will transmit in-band underlay uplink CDMA packets.

Experiment setup shown in Fig. 4.12 targets to evaluate performance of the shared band operation of LTE and CDMA-underlay IoT transmissions. In this setup, we have separate eNBs for CDMA IoT and LTE UEs. We are currently integrating IoT eNB and OAI/LTE eNB into one unified eNB. In our experiments, for CDMA IoT

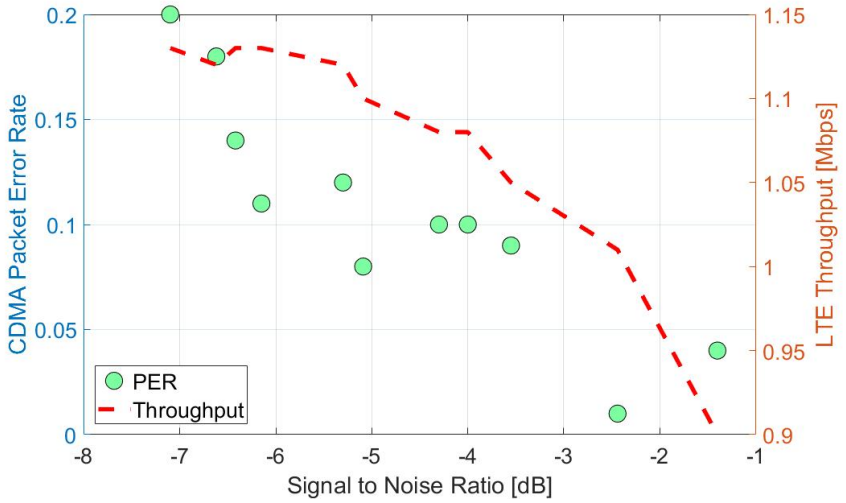


Figure 4.13: Throughput of the LTE transmission and Packet Error Rate(PER) of the CDMA-based IoT transmission as a function of the CDMA Signal-to-Noise Ratio(SNR).

transmission, packet payload and interval between packet transmission is kept constant at 15 Bytes and 60 millisecond, respectively. CDMA IoT SNR is varied as mentioned in Section 4.1.1. LTE UE transmits UDP traffic using iperf. LTE is not completely loaded and achieves 1.15 Mbps throughput without any CDMA transmission. As shown in Fig. 4.13, with simultaneous CDMA and LTE transmission, CDMA average SNR is observed to be in the range -7.1 dB to -1.4 dB and PER is in the range of 0.01 to 0.22 (due to not fully loaded LTE channel). As CDMA SNR increases, LTE throughput decreases upto a maximum of 20%. This scenario provides insight on design details of CDMA-underlay IoT protocol considering OFDMA structure of LTE.

Chapter 5

Related Work

In this chapter, we will discuss some waveforms and technologies that have been suggested to handle IoT traffic in the future 5G wireless communication systems. There have been waveforms and standalone systems proposed to satisfy the vast range of new services in the net generation internet. Some of the service requirements of 5G include Gbps downlink speeds, massive number of HTC (Human-Type Communication) and MTC (Machine-Type Communication) devices and ultra low latency. [30]

Some of the suggested waveform techniques for 5G are Generalised Frequency Division Multiplexing (GFDM) [31] which is a block based multi-carrier system similar to OFDM. But, instead of adding a cyclic prefix to every symbol, GFDM adds a single prefix to an entire slot (7 sub-symbols). The symbols are smaller in size and it improves the spectral efficiency. GFDM has significantly low PAPR and a better BER than OFDM, but it still does not provide us a platform for low latency applications. Filter Bank Multicarrier (FBMC) and Universal Filtered Multi-Carrier (UFMC) [32,33] are another set of proposals from the community. FBMC is a multicarrier system which instead of digitally filtering the complete band, filters on a per subcarrier basis. Instead of sinc-pulses, the subcarriers have a more suitable shape according to the filter design with reduce side lobe levels. With UFMC, filtering is applied on a per sub-band basis instead of subcarrier as in FBMC. UFMC is as spectrally efficient as FBMC, whereas FBMC suffers from high time domain overheads. FBMC also sufers from higher overhead for UL/DL synchronization. For the purpose of bursty small packet IoT transmissions, UFMC is a better fit [32] and can be used to enable low latency modes with low energy consumption. More waveform techniques are discussed in [34–36]. All the above mentioned techniques provide benefits like reduced PAPR, higher spectral efficiency,

Massive MIMO support, enabling a framework with low latency. One advantage that underlay CDMA has with respect to the aforementioned techniques are that it works in band with the current LTE architecture. The CDMA transmission a being random access technology provides significantly lower access and transmission delay.

Narrowband IoT (NB-IoT): As 3GPP is moving towards standardizing the 5G New Radio [37–40], Narrowband-IoT [12] and various RAN aspects for Machine-Type Communication [19, 41] are considered. However, the latency numbers for NB-IoT are expected to be much higher than LTE/LTE-A as the duration between narrowband PSS and SSS are significantly more. As seen in Section 2.3.1.1, the initial access delay is already high for LTE. The target RACH procedure latency for emergency messages in NB-IoT is 10 seconds. Also, NB-IoT allocates subcarriers from the existing spectrum used by LTE causing degradation in LTE throughput as well. Thus, LTE and NB-IoT have high scheduling overhead and access delay. Since NB-IoT uses RACH for newly arriving UEs, sudden surge in IoT traffic could result in high collisions.

The majority of the proposed standalone technologies are systems operating in sub-6GHz frequencies and do not intend to merge with cellular services. Weightless [42] is one such effort for Low Power Wide Area Network (LPWAN). It has an open standard, and nWave [43] has already developed a prototype for the Weightless-N standard. SigFox [11] is another standalone network solution for Internet of Things. It is ultra-narrowband and supports packets sizes of 12 bytes. LoRa Alliance [13] is another standalone option for IoT network, which is based on a proprietary spread spectrum developed by Semtech. These dedicated IoT network [44–46] suffer from scalability when simultaneous transmissions from IoT result in collision of packets followed by re-transmissions. Nonetheless, even in hypothetical scenario of abundance of chirp codes, often the allocated bandwidth is not enough to handle the surge of IoT traffic.

With the possibility of extending LTE-U in multiple unlicensed spectrum, it is envisioned that in recent future standalone IoT systems in unlicensed spectrum will suffer significantly due to interference from LTE. This defines the need to coexist with the cellular network, which can be achieved using underlay transmission techniques such as those proposed here.

Chapter 6

Conclusion

The massive deployment of IoT devices in the next few years will overwhelm the current 4G network and could also affect the future 5G network by causing a significant amount of surge in control plane signaling overhead and in network latency. This makes 4G unsuitable for IoT deployment and this motivated us to design a CDMA-based PHY/MAC protocol for IoT devices. The proposed IoT system coexists with in-band LTE operation and allows uplink sporadic IoT data transmission without a need for resource allocation from the LTE network. The CDMA-based IoT system is a random access protocol which experiences very low access delay compared to LTE. We implemented the proposed protocol using software-defined radio platform for exemplary scenarios as a proof-of-concept. We studied the CDMA based IoT system to understand its limitations. The main limitation of the CDMA based IoT system is the high transmission delay for large packet transmissions. As seen from our analysis and results, upto 13 IoT devices can transmit packets without affecting each other's performance in 1 MHz bandwidth. Depending on the requirements, the latency in CDMA can be controlled with various system parameters like bandwidth, modulation schemes, message sizes and spreading code length.

The underlay IoT transmissions in presence of an ongoing LTE transmission cause the LTE throughput to degrade a little. This degradation in throughput can be controlled in future by controlling the transmit power of IoT UEs. Coordination amongst OFDM (in LTE) and CDMA will be vital in managing the spectrum and allocating the resources better, leading to a better performing IoT system.

Appendix A

CDMA Code

The appendix describes the codes written for running Code Division Multiple Access (CDMA) on Ettus USRP Software Defined Radios B210/X310. The code is written first in MATLAB and then was moved to C to interface with the USRP in real-time. The C code is available on a private repository in bitbucket and also in the image *cdma-new.ndz* on ORBIT, WINLAB. The MATLAB code is available on Google Drive. The CDMA Receiver was run in a separate node, keeping it close to the LTE eNodeB and this setup can be seen in Section 4.1.2.

A.1 MATLAB code

The MATLAB code is not the complete implementation of CDMA and only validates the use of different algorithms and techniques. Once we figured out what algorithms we needed to use, a complete code was written in C/C++.

A.1.1 CDMA Transmitter

The CDMA Transmitter code is in the script *Transmitter.m*. The transmitter in MATLAB is different from the transmitter in C, as the packet created does not have fields for UE ID and Message Size. We will discuss the C code in later sections. The transmitter block diagram for the MATLAB code is given in Figure A.1. The transmitter generates CDMA packets and writes the samples to a file, which is then used to transmit over the USRP.

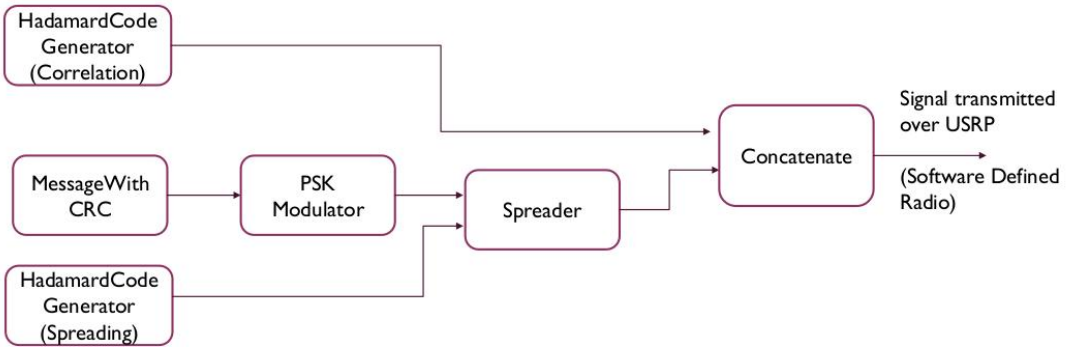


Figure A.1: CDMA Transmitter for MATLAB code

Different blocks/functions of the Transmitter are explained below:

1. **readFromFile:** This function takes two inputs, *FileName* and *NumberOfBytes* to be transmitted. The input file should contain a text with preferably *NumberOfBytes* characters and the value(or more) needs to be passed to the function as well. The function converts the characters to binary and appends a 16 bit CRC. For example, if the text is 10 bytes, it will be converted to a 96 bits codeword (80 bits for the message and 16 bits for CRC).

2. **bpskmod:** This function modulates the input signal or codeword using BPSK. Converts 1 to -1 and 0 to +1.

3. **HadamardGen:** This function generates a Hadamard code sequence. The inputs are *Length, Index, DataType, SamplesPerFrame.* It creates a Hadamard matrix of size Length x Length and gives out the sequence at the required Index.

4. **NormalizeGain:** This function multiplies the input sequence with the gain value as passed to the function.

5. **Spread:** This function spreads the input signal using the code sequence which is passed to the function. Every bit in the input signal is multiplied with the code sequence, also known as spreading.

6. **writetofile:** This function takes one signal as an input and converts them to IQ samples. For example, 2+3i is transmitted as 2 and 3 separately. The output

signal is thus twice the size of the input signal. These samples are then written to the file, and the filename is passed to the function. The samples are in the format int16 and that can be modified.

There are more functions which are not used in the code but could be useful:

1. **addawgn:** This function adds AWGN to the input signal and takes the required noise SNR value as an input.

2. **plotIQ:** This function takes two signals as inputs. First is the transmitted signal and the second is the received signal. Both need to be of the same size. The function gives the I-Q plot of the received samples depending on if the transmitted values are positive or negative.

A.1.2 CDMA Receiver

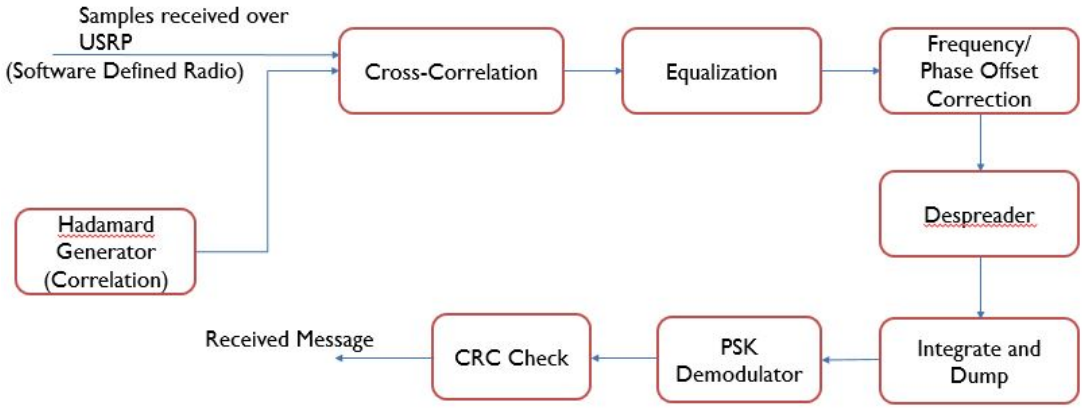


Figure A.2: CDMA Receiver for MATLAB code

The receiver reads IQ samples from a file and then processes it to find CDMA packets in the samples received. Functions in the CDMA receiver are explained below:

1. **readfromfile:** This function reads samples from the file passed as an input. The samples are then converted back to A + Bi format from A and B received separately.

2. **correlate:** This is the function where the cross-correlation is done to detect the

CDMA packets by looking for the preamble in the received signal. The function inputs are: Received Signal, Transmitted Packet(to fetch the size of the packet sent) and the Preamble. The received signal is large in size and is divided into blocks and these blocks are treated separately to look for the CDMA Packets/Preamble. The function returns the 2D array, called *corr*, which contains the contents of all the packets and *num* is the number of packets detected.

Important variables:

- arg: This value is the number of blocks you want to divide the received samples in. For better performance, select the number of blocks such that the block size is close to the size of the CDMA packet transmitted.
- thresh: This is the threshold value, using which the packet detections are done. The threshold value needs to be manually entered such that you only detect and process the actual CDMA packets and ignore the noise. This can be done by trial-and-error method by trying a few different values for the threshold when the number of packets transmitted is known.

3. **equalizeCorr:** This function equalizes the packet using the preamble as a training sequence. It uses Recursive Lease Square Estimation (RLS) to do the equalization. The parameters of RLS equalizer might need playing with depending on different scenarios. The output of this function is the samples without the preamble and just the spread packet. Source : `https://www.mathworks.com/help/dsp/ref/dsp.rlsfilter-class.html`.

4. **CS4:** This function does the Carrier Offset Correction using the *Carrier Synchronizer* function from MATLAB. It performs COARSE and FINE synchronization both. The source is `https://www.mathworks.com/help/comm/ref/comm.carriersynchronizer-class.html`.

   There are functions CS, CS2 and CS3 in the folder which perform Carrier Offset Correction in different ways.

5. **despreader:** This function is the despreader and performs exactly opposite to

the spreader. It divides every N samples, where N is the size of the PN code sequence with the PN code sequence.

6. **intdump:** This function integrates(adds) every N samples, where the N is the size of the PN Sequence and is passed as an input to the function.

7. **bpskdemod:** This function demodulates the input signal using BPSK. It converts negative values to 1 and positive values to 0.

8. **readSignal:** This function reads the final codeword and prints the character representation of the bits received. The CRC is first calculated and compared with the received CRC to know if the packet has been received in error or not. It prints out the character representation if the packet has been correctly received, else it is in error. The output is an array of 0s and 1s, where 0 is for no error and 1 for erroneous packets.

9. **calculatePktErrorRate:** This function calculates the packet error rate. It takes inputs as the total number of packets detected and the error array generated by the readSignal function above.

A.2 C/C++ code

A.2.1 Transmitter

The CDMA Transmitter code is written in C++. The block diagram of the CDMA transmitter as given in Figure A.3 in C++ is a little different from the one written in MATLAB.

Various functions used in the main, in order of which they are used:

- **readText**: Reads text from the user and stores it in the *msg* variable. The length of the characters is in *totalmsglength*. It also initializes a few variables and the function takes no arguments.

- **formPacket**: Takes the length of the packet as an argument. Creates the packet based on the length passed and copies the final bits to *txsignal*. This is the main

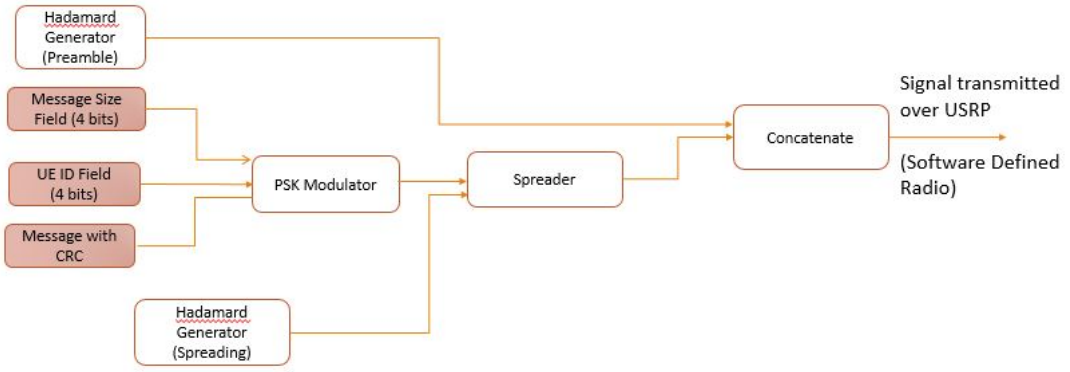


Figure A.3: CDMA Transmitter for the C++ code

function where all the packet formation happens at the transmitter. Following are the steps of forming the packet :

1. The first step is to create the *sendmsg* variable which only stores the part of the message (if we are varying lengths) that is to be sent. For example, if the input message is of size 15 characters, but we are sending only 10 characters. Then, *sendmsg* stores the 10 characters of the message.
2. We assign the UE ID to be the spreading code index for simplicity. Say, the spreading code index is 9, then we set the UE ID to be 9.
3. Functions **decToBinSize** and **decToBinID** take the message size and the UE ID as arguments and converts the decimal value into binary. The size field binary is stored in variable *sizeField* and ID is stored in *UEID*.
4. The text message from *sendmsg* is then converted to binary. CRC is calculated using CRC-16 checksum by calling the function **crcSlow**, which is present in *crc.c*. Message is stored in *msgBin* and calculated CRC is in *crcBin*. Length of the *msgBin* array containing the binary bits is *len* and length of the *crcBin* is *crclen*.
5. Total length, *totLen* is defined to be sum of message length, CRC length, sizeField in bits and UE ID in bits. The *codeword* is formed by copying all the bits to it. This is the packet that will be modulated and spread now.
6. Function **bpskmod** modulates the *codeword* using BPSK.

0 is -1

1 is +1

7. Fetches the spreading codes and the preamble code from the functions **getIndexCor** and **getIndexSpr**. The Hadamard matrix is generated in the **readText** function and the indices are fetched using these functions.
8. Packet spreading is done in the function **spreading** and the spread length is *packetLength*. The spreading is done depending on what the UE ID is, since that determines the spreading code index.
9. Preamble is added to the packet and is multiplied by a factor of 32767 to increase the gain of the packet transmitted. The final packet is in *txsignal*.

- The do-while loop checks for the *stop_signal_called* and the *repeat* variable to know if there are more packets that need to be sent. It passes the appropriate data type to the template of **send_cdma** function, depending on what the message type is. We use **short** but can use anything.

  send_cdma transmits the packet by copying the packet data from *txsignal* to the *tx_stream* only *buff.size()* samples at a time. *buff.size()* is equal to *samps_per_buff*.

- **formPacket** can be used inside the while loop if we want to vary the size of the packets transmitted. Delay (stored in *delay* variable) between the packets is controlled by making the thread sleep for some time/random times. If the packets sent is equal to the *pktCount*, then it comes out of the loop.

A.2.1.1 How to run CDMA Transmitter

The README in the CDMA folder can be referred to understand how to run the CDMA transmitter. The instructions are: Go to /uhd/host/build/examples. It should have tx_cdma and rx_cdma executables.

1. tx_cdma.cpp sends random sized packets at random intervals. Intervals being multiples of the delay(in milliseconds) given in the command line.

2. To run the CDMA Transmitter code (random packet sizes and random delay):

   ./tx_cdma –freq 2560e6 –rate 1e6 –gain 90 –repeat –delay 100 –UE 2 –pkt 100

   –freq : Center frequency (in Hz)

   –rate : Sampling Rate (in Hz)

   –gain : Gain

   –delay : Delay (in milliseconds)

   –UE arg (=2) : UE Identification Number (default = 2)

   –pkt arg (=100) : Packet count (default = 100)

.

A.2.2 Receiver

The CDMA Receiver code is written in C. Instructions on how to run the receiver is in the README file. The block diagram of the CDMA Receiver is in Figure A.4. The flow diagram of the CDMA Receiver can be seen in Figure A.5.

The CDMA Receiver is running all the time and is constantly receiving samples from the USRP. It looks to find the CDMA Packets in the samples it receives using cross-correlation. Further processing is done, to find the UE ID, the size of the packet and then decode the message accordingly.

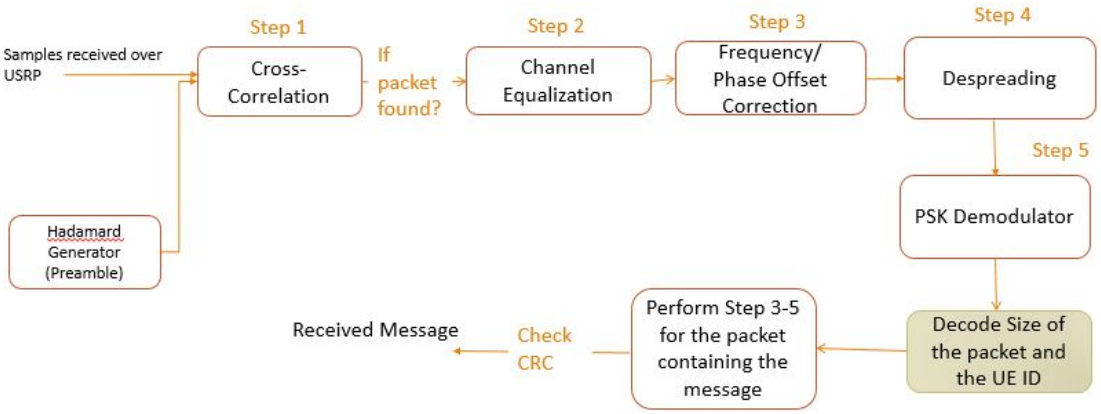


Figure A.4: CDMA Receiver for the C code

At the receiver, there are two threads running simultaneously. One is the main thread, which reads the samples from the USRP and looks for the packets in every set

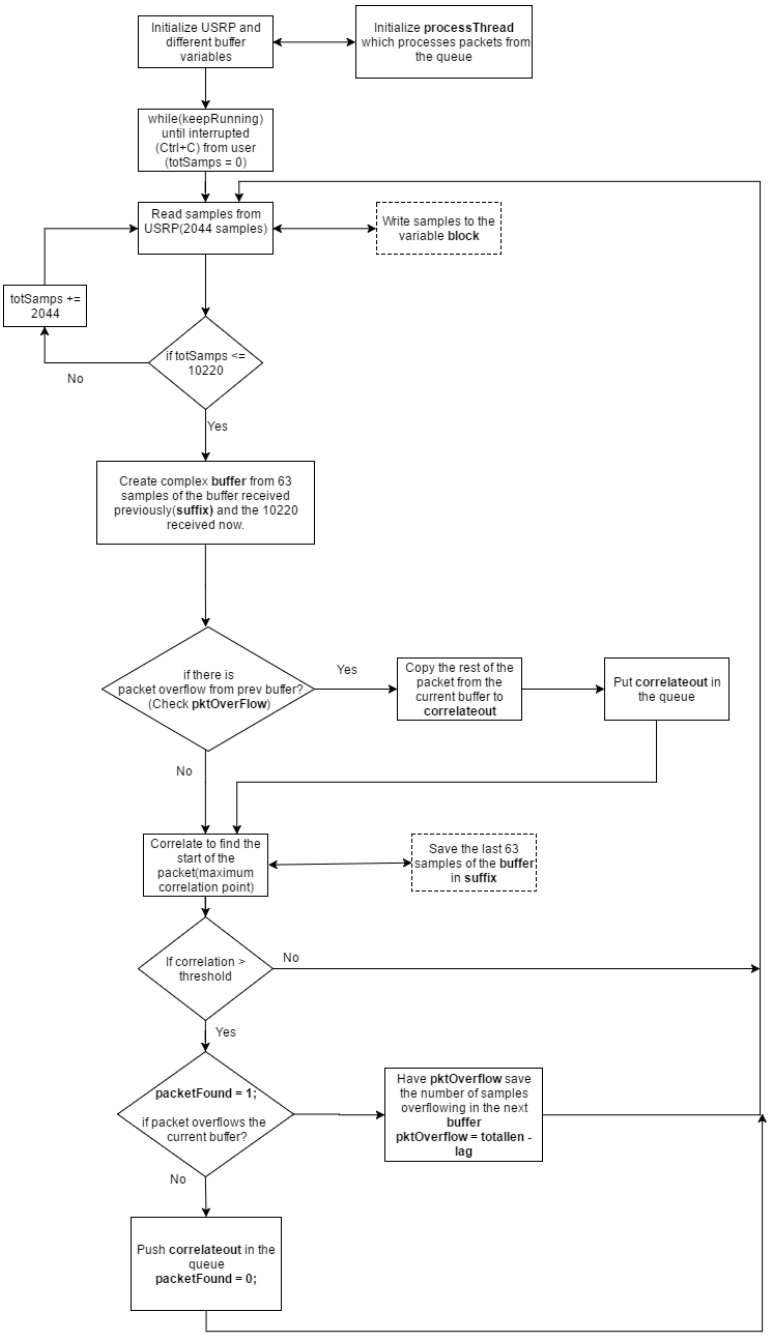


Figure A.5: Flow Diagram of the CDMA Receiver

of received samples. If the thread detects a packet, it puts the packet in an unlimited sized Queue and increases the queue's size. The other thread is the one which monitors the Queue. If the queue is empty, it keeps checking the queue's length to see if a packet has arrived. If there is one or more elements in the queue, it takes out a packet from the front of the queue (First in, First Out). It dequeues the packet and further processing is done to extract the message out. This processing is done in the processThread as explained in A.2.2.2.

A.2.2.1 Main Thread

Steps in the main thread :

1. The main thread/function initializes some basic variables to get the USRP setup. It sets up the USRP's streamers,etc and more can be understood about it on UHD website.

2. *packetMax* is a variable that stores the maximum bits that the message can have. The maximum size of our messages are 15 bytes. Maximum characters = 15 bytes = 120 bits. CRC = 16 bits. Total = 136 bits. This can be increased or decreased to any value. For ex, packetMax = 2056 bits for max = 256 bytes.

3. Initializing variables to create the **processThread**.

4. Sets up the Hadamard code matrix. Initialize the *corrindex*, *spreadindex(n)*, which are the preamble code index and the spreading code index(s).

5. **pthread_create** creates the **processPacket** thread.

6. while loop is running until it is externally interrupted by the user using Ctrl + C. In this loop, samples are read from the USRP and are dumped into a file.
   The array *buffer* stores the samples from *block* and prefixes it with M-1 samples of the previous buffer. The last M-1 samples of the previous buffer is stored in the variable *suffix*.

7. *correlateout* is the array which stores the packets as they are detected. If the packet is overflowing from the previous buffer, we copy *pktOverflow* samples from

the new buffer to *correlateout*. This completes our packet and we enqueue the packet to the queue for processing. We set *pktOverflow* to -1.

8. **correlate** is the cross-correlation function where the preamble detection is done by cross-correlating the received buffer with the preamble. The packet is stored in *correlateout*. If we find a packet(*packetFound = 1*) and the packet is not overflowing(*pktOverflow = -1*), then we enqueue the packet and reset the variables for the next packet.

   The packet detection is done by setting a threshold. The threshold decides if the packet we are "detecting" is noise or the actual expected packet. For this, we run the CDMA receiver without any CDMA transmissions. This gives us an idea on what the noise level is by giving the cross-correlation values of the preamble with the noise. Then, setting the threshold for cross-correlation accordingly (more than the noise level) to detect actual packets.

   This part can be modified to include multiple packet detections in one processing buffer. Currently, it only detects one packet in one buffer.

9. Once we exit the while loop, the SNR(Signal to Noise Ratio) and PER(Packet Error Rate) is calculated. These statistics are printed out to the user.

A.2.2.2 Process Thread

- Outer while loop runs until interrupted externally by the user using Ctrl + C.
- The inner while loop keeps a check on the *queueSize* and runs while there exists an element in the queue. If there is an element in the queue, then it follows the following steps:

  1. Get the packet from the front of the queue and remove it from the queue. This is stored in variable *pkt*.
  2. The packet is processed first to get the size of the packet and the UE ID. The despreading is done using *spreadindex1* and if the UE ID matches the *spreadindex1*, then the message is from UE ID = *spreadindex1*. If not, we

despread using *spreadindex2* and then the UEID should match the *spreadindex2*. This is how we do multi-user detections at the CDMA Receiver. If none of these hold true, we discard the packet as it is not from any of the two UEs.

The processing involved equalization, frequency/phase offset correction, demodulation, despreading and converting from binary to decimal/characters. The functionalities of these are the same as it was in the MATLAB code, so we are not going into detail again. Refer Section 2.2 for more details.

3. Once we have retrieved the message size and the UE ID, the next step is to process the packet to get the message out. *packetOffset* gives us the offset of where the spread message starts. (= M + (L*sizeBits) + (L*IDBits))
4. Depending on what the *UEID* is, the despreader code is chosen and the message is finally retrieved in **readSignal** function.
5. Similar to the MATLAB function, **readSignal** calculates the CRC checksum of the bits received and compares it to the received CRC. If they are the same, there is no error in the reception and it prints out the received message after converting bits to character.

A.2.2.3 How to run CDMA Receiver

The details can be found in the README. The instructions are also given here:

- rx_cdma.c receives the samples, stores them in a buffer (of n size, which is a parameter given at runtime in the command line) and then processes it to find CDMA packets. It uses COARSE and FINE Carrier Frequency offset estimator and corrector . It discards packets with CRC mismatch.
- To run the CDMA Receiver code: ./rx_cdma -f 2560e6 -r 1e6 -g 20 -n 10220 -t 300
  - -n: Samples buffer size.(For X310s , say 36400 for simplicity since 364 is the buffer size. For B210s, the buffer size is 2044, then make -n as 10220)

  - -t: threshold for correlation output. To calculate the threshold (at a particular gain value), First run the receiver without any ongoing transmissions.

- ./rx_cdma -f 2560e6 -r 1e6 -g 20 -n 10220

  After stopping the program, we see "Maximum correlation peak" give out a value. This is the maximum value which the cross-correlation gives out with noise.

  We need to set the threshold more than that so that we do not detect any incorrect start of the CDMA packet. Now, say if the Maximum correlation peak is 200. We can select the threshold to be 250, so that we do not have any incorrectly detected packets.

  - -f: Center frequency in Hz.
  - -r: Sampling Rate in Hz.
  - -g: Gain
  - -o : output filename (default = test.dat)

References

[1] S. Mathur, D. Saha, and D. Raychaudhuri. Poster: Cross-layer MAC/PHY protocol to support IoT traffic in 5G. In *ACM Mobicom*, 2016.

[2] S. S. Sagari, Siddarth Mathur, D. Saha, S. O. Amin, R. Ravindran, I. Seskar, D. Raychaudhuri, and G.Q. Wang. Realization of cdma-based iot services with shared band operation of lte in 5g, 2017. ACM SIGCOMM 2017 Workshop on Mobile Edge Communications (MECOMM2017).

[3] 3GPP TS 36.213 version 13.0.0 Release 13. Lte; evolved universal terrestrial radio access (e-utra); physical layer procedures.

[4] C. H. Wei, G. Bianchi, and R. G. Cheng. Modeling and analysis of random access channels with bursty arrivals in ofdma wireless networks. *IEEE Transactions on Wireless Communications*, 14(4):1940–1953, April 2015.

[5] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016 –2021 . `https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html`.

[6] The Internet of Things: How the Next Evolution of the Internet Is Changing Everything . `https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf`.

[7] Report: Popular Internet of Things Forecast of 50 Billion Devices by 2020 Is Outdated.

[8] Gartner IoT Forecast . `http://www.gartner.com/newsroom/id/3598917`.

[9] M.Z. Shafiq, Lusheng Ji, A.X. Liu, J. Pang, and Jia Wang. Large-scale measurement and characterization of cellular machine-to-machine traffic. *Networking, IEEE/ACM Transactions on*, 21(6):1960–1973, Dec 2013.

[10] S. Mathur, S. S. Sagari, S. O. Amin, I. Seskar, R. Ravindran, D. Saha, D. Raychaudhuri, and G. Wang. Demo abstract: CDMA-based IoT services with shared band operation of LTE in 5G, 2017. IEEE Infocom Workshop 2017.

[11] Sigfox. `http://www.sigfox.com`.

[12] Y. P. E. Wang, X. Lin, A. Adhikary, A. Grovlen, Y. Sui, Y. Blankenship, J. Bergman, and H. S. Razaghi. A primer on 3gpp narrowband internet of things. *IEEE Communications Magazine*, 55(3):117–123, March 2017.

[13] Lora. `http://www.lora-alliance.org`.

[14] 3GPP LTE. http://www.3gpp.org/technologies/keywords-acronyms/98-lte.

[15] Vijayalakshmi Chetlapalli Deepti Singhal, Mythili Kunapareddy. Lte-advanced: Latency analysis for imt-a evaluation.

[16] J.G. Proakis and M. Salehi. *Communication Systems Engineering*. Pearson Education. Prentice Hall, 2002.

[17] K. S. Gilhousen et al. On the capacity of a cellular CDMA system. *IEEE Transactions on Vehicular Technology*, 1991.

[18] 3GPP TS 36.331 version 13.0.0 Release 13. Lte; evolved universal terrestrial radio access (e-utra); radio resource control (rrc).

[19] 3GPP TR 37.868 version 11.0.0. 3gpp; technical specification group radio access network; study on ran improvements for machine-type communications;.

[20] 3GPP TR 45.820 V13.1.0 (2015-11). Cellular system support for ultra-low complexity and low throughput internet of things (CIoT).

[21] Zhao Liu and Magda El Zarki. Performance analysis of ds-cdma with slotted aloha random access for packet pcns. *Wirel. Netw.*, 1(1):1–16, February 1995.

[22] D. Raychaudhuri and K. Joseph. Performance evaluation of slotted aloha with generalized retransmission backoff. *IEEE Transactions on Communications*, 38(1):117–122, Jan 1990.

[23] D. Raychaudhuri. Performance analysis of random access packet-switched code division multiple access systems. *IEEE Transactions on Communications*, 29(6):895–901, Jun 1981.

[24] D. Raychaudhuri and K. Joseph. Performance evaluation of slotted aloha with generalized retransmission backoff. *IEEE Transactions on Communications*, 38(1):117–122, Jan 1990.

[25] D. M. Grieco. The capacity achievable with a broadband cdma microcell underlay to an existing cellular macrosystem. *IEEE JSAC*, 1994.

[26] S. Sagari et al. Coordinated dynamic spectrum management of LTE-U and Wi-Fi networks. In *IEEE DySPAN*, Sept 2015.

[27] M. Luise and R. Reggiannini. Carrier frequency recovery in all-digital modems for burst-mode transmissions. *IEEE Transactions on Communications*, 43(2/3/4):1169–1178, Feb 1995.

[28] Michael Rice. *Digital communications: a discrete-time approach*. Pearson Education India.

[29] N. Nikaein et al. Openairinterface: A flexible platform for 5G research. *SIGCOMM*, 2014.

[30] Ngmn alliance, "5g white paper, february 2015, available at. https://www.ngmn.org/uploads/media/NGMN_5G_White_Paper_V1_0.pdf.

[31] G. Fettweis, M. Krondorf, and S. Bittner. Gfdm - generalized frequency division multiplexing. In *VTC Spring 2009 - IEEE 69th Vehicular Technology Conference*, pages 1–4, April 2009.

[32] F. Schaich and T. Wild. Waveform contenders for 5g x2014; ofdm vs. fbmc vs. ufmc. In *2014 6th International Symposium on Communications, Control and Signal Processing (ISCCSP)*, pages 457–460, May 2014.

[33] F. Schaich, T. Wild, and Yejian Chen. Waveform contenders for 5g - suitability for short packet and low latency transmissions. In *Vehicular Technology Conference (VTC Spring), 2014 IEEE 79th*, pages 1–5, May 2014.

[34] S. Y. Lien, S. L. Shieh, Y. Huang, B. Su, Y. L. Hsu, and H. Y. Wei. 5g new radio: Waveform, frame structure, multiple access, and initial access. *IEEE Communications Magazine*, 55(6):64–71, 2017.

[35] B. Farhang-Boroujeny and H. Moradi. Ofdm inspired waveforms for 5g. *IEEE Communications Surveys Tutorials*, 18(4):2474–2492, Fourthquarter 2016.

[36] F. Yang and X. Wang. A novel waveform for massive machine-type communications in 5g. In *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–5, March 2017.

[37] C. Sexton, N. J. Kaminski, J. M. Marquez-Barja, N. Marchetti, and L. A. DaSilva. 5g: Adaptable networks enabled by versatile radio access technologies. *IEEE Communications Surveys Tutorials*, 19(2):688–720, Secondquarter 2017.

[38] 3GPP TS 37.340 V0.2.0. Lte; evolved universal terrestrial radio access (e-utra) and nr multiconnectivity.

[39] 3GPP TR 38.801 V14.0.0. Study on new radio access technology: Radio access architecture and interfaces.

[40] 3GPP TR 38.804 V14.0.0. Study on new radio access technology: Radio interface protocol aspects.

[41] 3GPP TR 37.869 V1.0.0. Study on enhancements to machine-type communications (mtc) and other mobile data applications; radio access network (ran) aspects. 2017.

[42] Weightless. `http://www.weightless.org`.

[43] Nwave. `http://www.nwave.io`.

[44] Claire Goursaud and Jean-Marie Gorce. Dedicated networks for IoT : PHY / MAC state of the art and challenges. *EAI endorsed transactions on Internet of Things*, October 2015.

[45] W. Yang, M. Wang, J. Zhang, J. Zou, M. Hua, T. Xia, and X. You. Narrowband wireless access for low-power massive internet of things: A bandwidth perspective. *IEEE Wireless Communications*, 24(3):138–145, 2017.

[46] X. Xiong, K. Zheng, R. Xu, W. Xiang, and P. Chatzimisios. Low power wide area machine-to-machine networks: key techniques and prototype. *IEEE Communications Magazine*, 53(9):64–71, September 2015.