Performance of a Frequency-Hopped Real-Time Remote Control System in a Multiple Access Scenario

by

Frank Cervantes

M.Sc. (2004)

A thesis submitted to the

Faculty of Graduate Studies and Research

in partial fulfillment of the requirements for the degree of

Master of Applied Science

Ottawa-Carleton Institute for Electrical and Computer Engineering

Department of Systems and Computer Engineering

Carleton University

Ottawa, Ontario, Canada

September, 2010

The undersigned recommend to
the Faculty of Graduate Studies and Research
acceptance of the thesis

Performance of a Frequency-Hopped Real-Time Remote Control System in a Multiple Access Scenario

submitted by

Frank Cervantes

M.Sc. (2004)

In partial fulfillment of the requirements for
the degree of Master of Applied Science in Electrical Engineering

_____

Chair, Howard Schwartz, Departement of Systems and Computer Engineering

_____

Thesis Supervisor, Prof. Marc St-Hilaire

Carleton University
September, 2010

Abstract

A recent trend is observed in the context of the radio-controlled aircrafts and automobiles within the hobby grade category and Unmanned Aerial Vehicles (UAV) applications moving to the well-known Industrial, Scientific and Medical (ISM) band. Based on this technological fact, the present thesis evaluates an individual user performance by featuring a multiple-user scenario where several point-to-point co-located real-time Remote Control (RC) applications operate using Frequency Hopping Spread Spectrum (FHSS) as a medium access technique in order to handle interference efficiently. Commercial-off-the-shelf wireless transceivers ready to operate in the ISM band are considered as the operational platform supporting the above-mentioned applications. The impact of channel impairments and of different critical system engineering issues, such as working with real clock oscillators and variable packet duty cycle, are considered. Based on the previous, simulation results allowed us to evaluate the range of variation for those parameters for an acceptable system performance under Multiple Access (MA) environments.

Acknowledgement

To my parents, who encouraged me to pursue a better status in my professional life, and provided me with constant spiritual support and love: this is for you, with eternal gratitude and love.

I would like to express my sincere gratitude to my thesis Supervisor, Professor Marc St. Hilaire, for his excellent guidance, moral support, and immense patience throughout this work, and during my degree in general. It was a valuable experience for me as a professional to share ideas with you, especially when facing certain difficult technical situations during the stages of this research work.

I would like to extend my thanks to Professor Ioannis Lambadaris for bringing to this project his wise suggestions, his pragmatic vision, and his enormous knowledge in a wide range of topics in radiocommunications and control systems. I deeply appreciate his willingness to provide answers to all the questions I had during those difficult moments commonly faced in research work.

Finally, to those friends who contributed with their support, ideas, and work, I express my appreciation. Thank you, Joel Lugo, for sharing your knowledge in programming techniques, for being a good listener every time a discussion about this topic came up, and for encouraging me to make this thesis real. Thank you, Ania Portales for your moral support and sincere friendship. Thank you, Christian Crouse, for your patience and time dedicated to the advising of the written part of this thesis.

Table of Contents

List of Tables

List of Figures

List of Acronyms

ACI	Adjacent Channel Interference
AFHSS	Asynchronous Frequency Hopping Spread Spectrum
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BFSK	Binary Frequency Shift Keying
CDMA	Code Division Multiple Access
CMOS	Complementary Metal-Oxide Semiconductor
DLL	Delay-Locked Loop
DSSS	Direct Sequence Spread Spectrum
FAP	Frequency Agile Protocol
FCC	Federal Communication Commission
FDMA	Frequency Division Multiple Access
FH	Frequency Hopping
FHSS	Frequency Hopping Spread Spectrum
FIFO	First In First Out
FSK	Frequency Shift Keying
FSM	Finite State Machine
HRT	Human Response Time
ISM	Industrial, Scientific, Medical
MA	Multiple Access
MAI	Multiple Access Interference
MCU	Microcontroller Unit
PER	Packet Error Rate
PLL	Phase-Locked Loop
PTX	Primary Transmitter Device
PRX	Primary Receiver Device
RC	Remote Control
RCU	Remote Control Unit

RF	Radio Frequency
RSSI	Radio Strength Signal Indicator
SLOP	System Lag Occurrence Probability
SoC	System on Chip
SPI	Serial Peripheral Interface
SR	Stored Reference
SFHSS	Synchronous Frequency Hopping Spread Spectrum
SNIR	Signal to Noise plus Interference Ratio
SNR	Signal to Noise Ratio
TDL	Tau-Dither Loop
TDMA	Time Division Multiple Access
ToA	Time on Air
TR	Transmitted Reference
UAV	Unmanned Aerial Vehicle

Chapter 1

Introduction

The operation of radio-controlled aircrafts and cars within the hobby grade category in the well-known and license-free Industrial, Scientific and Medical (ISM) frequency band have become more and more popular in the last few years [1], [2]. Many planes can be now operated in the same physical area without worries about frequency control, eliminating the need to check everyone else's channel numbers prior to flying. The possibility of turning up at a club with the wrong crystal in the transmitter unit does not constitute a worry anymore. Operating at 2.4 GHz also puts the radio control out of the frequency range of any noise caused by the other electronic components that are part of the aircraft, such as the motor, speed controller, and any metal-to-metal noise eliminating interference and glitching that can affect traditional frequency Remote Control (RC) systems.

In this chapter, we will first present some background information or update concerning the state-of-the-art of the targeted real-time RC applications mentioned above (i.e., low-power ISM-based RC applications, which implies the use of well-known ISM transceivers). The problem statement that has motivated this research work is then set, followed by the research objectives that need to be accomplished in order to produce a complete solution to the former problematic situation. Finally, main contributions of the present work to the status of the knowledge in the topic are delivered.

1.1 Background and Motivation

The aforementioned extraordinary trend that has emerged from the traditional RC systems implementation is widely supported by the hardware electronics point of view, since a plethora of powerful wireless low-power System on Chip (SoC) in the ISM band (popularly known as ISM transceivers)is already available in the market at reasonable costs [3-6].

SoC ISM transceivers have evolved continuously in time through five generations; the current generation runs at a high level of system integration. It is common not only to find a microcontroller (introduced in 2006 within the fourth generation) that is already embedded on a chip and some other well-established data link layer capabilities such as the Enhanced ShockBurst™ protocol, but also fully-programmable frequency-agile synthesizers with a Phase Lock Loop (PLL) settling time as low as 90 μs [3], [4]. Examples of the fifth generation transceivers are the nRF24LU1 [7] and the CC2511F32 [8] from Nordic Semiconductor and Texas Instruments Inc., respectively.

Three techniques have evolved in order to allow for systems coexistence in the ISM band: Time Division Multiple Access (TDMA), Direct Sequence Spread Spectrum (DSSS), and Frequency Hopping Spread Spectrum (FHSS). However, it has been shown that FHSS is the more suitable technique for packet-based real-time RC applications constrained to relatively low data-rate and low power [4]. The actual status of the U.S sub-1GHz ISM band, covering from 902 to 928 MHz, is well-known as being a shared spectrum resource with many license-free radio communication systems that interfere with one another [9]. FHSS has emerged as one of the variants of spread spectrum technique which enables coexistence of multiple networks (or same network devices) within the same physical area [10]. In this sense, the Federal Communication Commission (FCC) recognizes FHSS as one of the techniques withstanding “fairness” requirements for unlicensed operation in the ISM band.

Based on what has been previously mentioned, the present work is focused on the FHSS shared-spectrum access scheme *which has been successfully* employed in the realm of real-time RC applications [4].

When considering real-time remote-controlled applications, system latency is of critical importance and becomes our main constrain. Servomechanisms located on such an apparatus trigger physical movements on its control surfaces that are supposed to act proportionally to a given manual action performed at the Remote Control Unit (RCU). It will be assumed throughout this work that with the transmission of a single packet, all the

control commands needed by the system to drive its outputs are interchanged between the transmitter and the receiver. Radio channel is prone to errors, interference being the most important cause in this context at the time of addressing system latency. In the event of a considerable packet loss, control commands that were sent to the intended receiver are consequently lost and the remote-controlled unit becomes a non real-time follower of the RCU. In this case, a lag may happen during the radio control session if the time delay of system response exceeds certain real-time threshold.

Given all these notions, the problem statement is presented next.

1.2 Problem Statement

In previous research work such as [11], the performance evaluation of a *single* real-time RC application (i.e., a single master-slave association where no network environment is considered) operating in the ISM band under the effect of typical channel impairments has been addressed. Co-channel interference and channel noise effects were taken into account by means of the Packet Error Rate (*PER*) which is more suitable when dealing with packet-based radio systems. However, there still exists the need for a more realistic scenario (i.e., a multiplicity of users running identical RC applications) where a single system performance characterization is to be attained. Based on this primary idea, the main problem that motivated the present research can be fully stated as follows: a real-time RC system that operates in the ISM band and which is the basic constituent of a non-centralized control network (no network timing or system controller is considered mediating the access to the medium) that comprises a number of identical master-slave associations needs to be evaluated in terms of its performance.

The network, as shown in Figure 1.1, is basically a set of *unidirectional* links between paired nodes in a peer-to-peer fashion where the sending node, called here Primary Transmitter and denoted as (PTX), should be understood as an application (consider a remote control unit) running on a host controller connected to a typical ISM sub-1GHz or 2.4GHz transceiver.

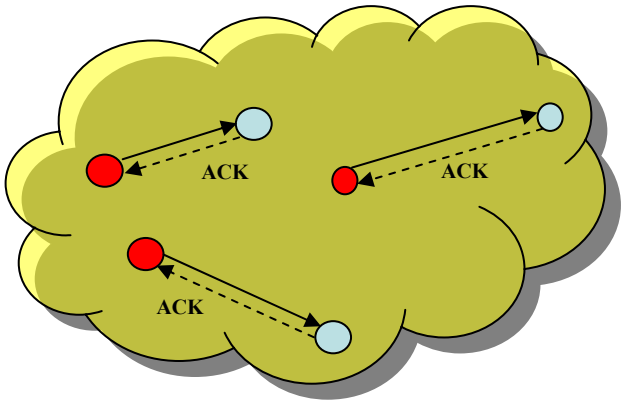


Figure 1.1 Typical RC FHSS-MA network configuration

The control unit is supposed to send commands with certain latency to a remote controlled flying apparatus where the Primary Receiver (PRX) or the intended receiver resides. Each association will interchange packets synchronously at a rate called *message rate* (*Tc*) as shown in Figure 1.2.

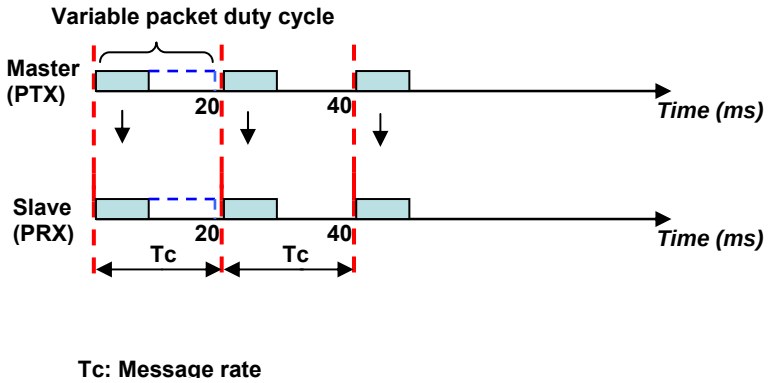


Figure 1.2 Basic way of communication that takes place in the targeted network

Each of these above mentioned associations is based on a *Stored Reference* (SR) slow FHSS system, which implements a uniform serial acquisition scheme enabled by a matched filter. The system performance is to be evaluated based on two metrics: System Lag Occurrence Probability (*SLOP*) considered in this work as the main metric and system throughput. They are considered satisfactory enough in estimating the behaviour of the system latency and its relationship with system design parameters.

System engineering issues such as the *non-synchronization between active users*, *clock drift and its impact on the correlation property of the hopping pattern* are to be analyzed within this proposed network environment. All the aforementioned issues that are intrinsic to the communication medium besides the previously cited system imperfections should bring us to a more realistic vision with regard the targeted application.

1.3 Research Objectives

Based on the problem statement that has been formulated in the previous section, the research objectives are the following:

1. Develop what was implemented in [11] by extending it to a multiple user scenario. In other words, a Synchronous Frequency-Hop Spread Spectrum (SFHSS) network with ideal clock oscillator is to be considered.
2. Compare results with those published in [11] in order to evaluate the present implementation.
3. Extend the scope in the research topic by also considering the Asynchronous Frequency-Hop Spread Spectrum (AFHSS) case besides the aforementioned SFHSS scenario but under certain system engineering issues (to mention: Operation with an imperfect clock oscillator and a variable data packet duty cycle at the user/application level) as part of a more realistic simulation ISM environment.
4. Evaluate the performance of a single real-time remote control system under the above mentioned conditions and make proper recommendations.

1.4 Methodology

As an initial task to be developed in this research work, an extension of what was done in [11] was proposed as a first level approach objective. For this aim, a set of slow-SFHSS RC real-time applications/users are to be simulated. In order to accomplish this:

1. An exhaustive review of [11], its theoretical and practical foundations are to be performed.
2. As a result of the above, a model for a typical slow-FHSS RC application and the transmission medium will be implemented. The plurarity of FHSS

In order to validate our model with respect to [11], the behaviour of the main system performance metric (i.e., *SLOP*) with respect to key system engineering parameters will be estimated through a set of experimental tasks. These computer-based experiments will run based on the model kernel to be built in the present development and are to be conducted considering all the necessary constrains at this level.

At a higher level of complexity and pragmatism with respect to the previously mentioned, two interesting case studies are of concern for this research: an asynchronous FHSS network (featuring variable packet duty cycle) as it is considered the more general case in a CDMA scenario, and a synchronous FHSS network where imperfect clock oscillators and a variety of hopping patterns are considered. In order to achieve these new case scenarios, it will be necessary to implement (model) the following aspects:

1. The asynchronicity between active users/applications or hopping sequences. Under this new condition, a variable packet duty cycle or the amount of time the transmitter will reside on each channel will also be modeled.
2. Real and independent clock oscillators driving the hopping sequences within a SFHSS network. Under this condition the implementation of a special set of hopping patterns that holds a very attractive cross-correlation property will be of interest.

Finally, set of experiments are to be performed according to each of the above mentioned special cases subject to their respective constrains. It should allow us evaluating the impact of the intra-system interference on system performance by means of the *SLOP* and throughput referred to a single user/application.

1.5 Main Contributions

Our research could be considered as an extension of the scope proposed in [11]. More specifically, we have introduced the following new aspects:

1. Extend the model developed in [11] to a *multi-user environment*, given by a multiplicity of identical slow-FHSS users. This aspect responds to the need for a more realistic scenario to be evaluated. This fact is the most valuable contribution to the state-of-the-art in the topic.
2. Under this multiple access umbrella, two scenarios have been simulated and analyzed: SFHSS and AFHSS Multiple Access (MA)-based RC networks.
3. Packet duty cycle and clock drift variability and their impact on system performance for the AFHSS and SFHSS-MA cases, respectively.

Based on these contributions, we are currently working on a conference paper.

1.6 Thesis Outline

The rest of this thesis is organized as follows:

Chapter 2: A literature survey is presented in order to give a concise update of the state-of-the-art developments in the following topics of interest: real-time RC system performance characterization and modeling. Relevant works on Frequency-Hop Spread Spectrum medium access technique are also reviewed.

Chapter 3: This chapter is dedicated to bring some insights in the receiver and channel modeling framework that supports both the SFHSS and AFHSS-MA scenarios.

Chapter 4: Simulation results for the SFHSS and AFHSS-MA scenarios are presented and discussed.

Chapter 5: General conclusions are given based on the partial results obtained in the previous chapter. Future research directions are also proposed.

Chapter 2

Literature Review

In this chapter, literature resources related to aspects of main interest for this research are reviewed in order to establish a state-of-the-art for the topic. Main system engineering issues, such as FHSS radio technique and related issues, and system main performance metric are concisely examined. In the specific case of the FHSS technique, not only the classic approach is taken into account, but also a more pragmatic vision of the technique which is in accordance with low-power SoC applications.

Finally, key aspects related with the modeling of a typical RC FHSS association (i.e., PTX-PRX) and the transmission medium will also be reviewed.

2.1 Frequency Hopping Spread Spectrum Radio Technique

In a standard FHSS system the transmitter–receiver pair cyclically hops according to a known pseudo-random sequence or code throughout a frequency band (W_{ss}), which is subdivided into M RF channels or frequency bands $\{f_1, f_2, ..., f_M\}$. The hopping sequence will dictate the current carrier frequency to be synthesized (located at the center of each of these sub-bands) and on which the transmission is supposed to take place. The time interval the transmitter spends on each channel is commonly referred as the *dwell time*. This parameter is equivalent to the Time on Air (ToA), as is more commonly referred in the ISM low-power SoC literature.

Usually, the hopping channels have the same bandwidth (Δf) that is selected according to a specific application. Based on all of this, it is possible to define the system processing gain (Gp) [12], [13]:

$$G_p = \frac{\text{RF Bandwidth}}{\text{Message Bandwidth}} = \frac{M\Delta f}{\Delta f} = M \tag{2.1}$$

Equation (2.1) encloses the intrinsic advantage of using FHSS as a system compared to a single channel conventional system that would use a bandwidth (Δf) centered around a specific constant RF carrier all the time and where narrow band interference may cause the system performance to notably deteriorate. FHSS systems deal better with the narrow band interference phenomenon by continuously allocating the RF emission through a series of M disjoint channels which comprise the total net hopping bandwidth W_{ss}.

In general, the well-known SR FHSS scheme, which implies no explicitly transmission of the spreading code or sequence, is typically used for RC applications based on low-power SoC systems. Due to this, synchronization between the transmitter and the receiver in both time and frequency domain is to be achieved. FHSS technique can also be implemented for these applications as *slow* or *fast*, depending on the rate between the amount of modulation symbols that are transmitted and the number of hops the system performs [12], [14-16].

For the synchronization of the reference frequency-hop pattern produced by the receiver synthesizer with the incoming pattern, it is in general desirable for the receiver to be capable of obtaining synchronization by processing the received signal [15], [17].

There are two domains of uncertainty when considering synchronization between the transmitter and the receiver: time and frequency. In order to get the locally generated code phase synchronized with the incoming delayed version of it, two processes are accomplished at the receiver: *acquisition* and *tracking*. Particularly, the acquisition stage always takes place after a system is powered-on or when synchronization is lost due to the effect of external factors, mainly channel noise or interference.

Acquisition provides coarse synchronization by limiting the choices of the estimated values to a finite number of quantized candidates of timing and frequency offsets [15]. When system enters in acquisition, such a coarse alignment could take some considerable time (known as *acquisition time*) depending on the search scheme or algorithm used in relation with the quality of the transmission medium. Acquisition schemes are commonly

based on *serial* or *parallel* (also known as the matched filter technique) search strategies. Something in common with these two search mechanisms is a correlation process which gives an *a priori* idea on how similar the signals to be synchronized upon reception are. The acquisition process is normally concluded by a control subsystem in the receiver once the phase of the locally generated sequence is brought to within a fraction of a hop with respect to the incoming sequence [15]. After this condition is detected and verified, the tracking system is activated. Further details on parallel and serial acquisition schemes can be found in [12], [15], and [18].

Parallel search technique exhibits the fastest acquisition time because all possible code offsets are examined simultaneously. However, its implementation could be expensive since it implies the same number of matched filters as hopping carriers. Instead, a serial search is more commonly employed. In this kind of search engine, alignment trials or *cells* are consecutively performed [15]. The concept of cell is closely related to the two domains of uncertainty mentioned above (time and frequency or similarly, the phase of the frequency hopping pattern). If the result of certain tests applied on a cell is not satisfactory, the actual cell is rejected and a kind of search process is started by modifying the current phase of the local sequence and attempting to correlate it again with the incoming version (i.e., a new cell will be tested). Otherwise, acquisition or coarse alignment is declared and tracking phase is triggered.

A combination of a matched filter and a serial search scheme is found to be very attractive when dealing with long period sequences and faster acquisition times are required. In this case, the matched filter subsystem is conceived to operate in a way that it will enable the serial search engine once a special short synchronizing sequence is correctly detected [15]. This scheme is thought as a short well-known (by the receiver) sequence which is embedded in a longer frequency-hop pattern that is used to transmit the payload. This special synch sequence is typically sent without data modulation prior to the long one and is supposed to be detected by the matched filter [17], [19]. This acquisition scheme combines the fast detection capability of the parallel search with the simplicity (low cost) of the serial search.

The tracking phase or fine synchronization takes over only if the above-mentioned correlation process satisfied the synch condition during the acquisition stage [9], [15]. The tracking process itself involves continuous operation where a fine alignment between the received and local generated hopping codes takes place by means of a feedback loop. There are a few popular tracking mechanisms or strategies that are currently employed in practice [12], [20], [21]. However, the predominant form of tracking in frequency-hopping systems is provided by the *early-late-gate* tracking loop [15], [18].

It is in general desirable to have the receiver operating in tracking phase as long as possible and to diminish the time it takes trying to acquire synchronism (i.e., while acquiring coarse alignment). This again depends on certain system parameters, such as the search algorithm and obviously the quality of the medium used for communication which may not be deterministic in nature as it is for instance the case of wireless.

At this point, the classical approach for FHSS operation, specifically the synchronization process, has already been explored. However, it is considered *impractical* in the real-time RC applications considered in this work, where small systems that normally include microcontrollers are involved [4]. Within this more specific context, a constant look-up table is commonly implemented for hopping operation where the allowed channels are stored (see Figure 2.1) [4], [9], [10], [13]. As a consequence, the synchronization process happens in a more simplistic way. The table can be built using a softtool such as SmartRF Studio and then stored in the microcontroller's memory [4].

Index	Register Settings	Frequency Number
0	[010…110]	20
1	[110…100]	8
⋮	⋮	⋮
40	[111…000]	2

Figure 2.1 Typical look-up table for a FHSS application involving low-power ISM transceivers

A similar table needs to be associated with each transmitter-receiver pair, being unique within a CDMA network. As it can be seen, there is a one-to-one correspondence between the microcontroller register setting and a specific RF frequency to be generated.

The pseudo-randomness property of the hopping engine is typically guaranteed by means of the output value of a sub-routine (whenever invoked) that simply implements a random number generator based on a specific seed. The value at the output of the random generator then becomes a pointer to the table index [9].

Although the look-up table method is widely used, its memory requirements increase linearly with the number of frequencies considered for hopping [9]. This could lead to a trade-off situation with respect to the system performance, as this appears to be directly proportional to the number of available hopping frequencies. A solution to this drawback is implementing a less memory-consuming algorithm where no table is stored, but just the lowest frequency value in the band (for instance 902 MHz). Based on that, random offsets are generated in order to produce the whole hopping set [9].

When trying to acquire synch with the transmitter (within this context), the receiver modifies its hopping rate to a much slower value than normal (i.e., when the whole system is in tracking condition). This is something typical in low-power SoC applications [3], [4], [10]. Under this condition, the receiver dwell time will be in the order of the number of hopping channels times the transmitter dwell time. The opposite strategy is

also employed, in which the transmitter occupies a given channel for a time period much longer than the receiver does and then starts transmitting a recognizable training sequence (i.e., alternating binary pattern: 1010…).

During acquisition, the receiver will always look for *valid data* while scanning all the channels. The data validation is usually performed using standard software squelch methods, such as Manchester coding and the Relative Signal Strength Indicator (RSSI) function among others [4]. When the valid data condition is reached at the receiver, then it will start hopping synchronously with the transmitter at the normal rate.

The fact that both the transmitter and the receiver hop at a constant rate will significantly simplify the synchronization process in time domain (i.e., "when to hop" uncertainty) [9]. Since the hop rate is fixed, the number of bits that are sent per single carrier that is generated will also be constant. As a result, the receiver could simply count bits in order to set the hop instant. Bit-level synchronization is either handled by hardware or by using an oversampling algorithm. Further details in these synchronization techniques could be found in [22] and [23], respectively.

In order to keep the transmitter-receiver pair hopping synchronously once acquisition is reached (i.e., "where to hop problem"), the following choices are commonly applied [4], [10]:

1. Same seed driving the pseudo-random generators (pointer to table index) at both the transmitter and the receiver side.
2. Transmitter supplies the intended receiver with the current table indexes to be used (embedded on the payload in the packet) in advance.

The pseudo-random hopping sequence is a key point regarding the performance of a FHSS-CDMA system. Details on hopping sequence generation can be found in [16], [18], and [24]. Specifically, the behaviour of the cross-correlation property of a given set of hopping patterns is of main interest for our purposes. When it is considered for instance a set of hopping patterns that holds an attractive cross-correlation property

within a SFHSS-MA scenario, as long as synchronism between active users is kept, a low level of self-system interference and therefore more system throughput could be achieved. However, this is a difficult condition to be attained in practice due to system imperfections.

2.2 Hopping Patterns

From the literature addressing FHSS-MA or ISM low-power transceiver applications, it is customary that featured hopping patterns follow either a *general random stationary process*, under which Markov and Memoryless principles are considered for instance (see Figure 2.2), or a deterministic rule [25].

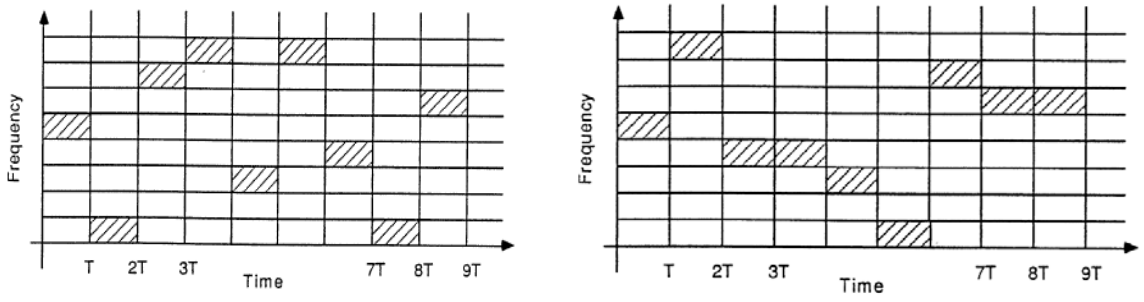


Figure 2.2 Markov (left) and Memoryless (right) based hopping patterns

Throughout this research work, we will implement both categories as part of our modeling framework. A class of a deterministic non-repeating hopping set that is touched in this section will be specially applied to a more realistic SFHSS-MA network scenario where imperfect clocks are considered.

In case of a hopping pattern following a Markov stationary process, it is only required that given the actual frequency number, the very next value to be generated should not be the same (i.e., $P(f_{j+1}^k = f_j^k) = 0$). Where (f_j) represents a given RF channel number to be used in the hopping sequence at time instant j^{th} by user k^{th}. This implies that: $P(f_{j+1}^k = v_n \mid f_j^k = v_r) = (q-1)^{-1}$ with $1 \le n, r \le q$ [25], [26]. The parameter q refers to the number of distinct hopping carriers to be used in the application. As it can be seen,

the selection criterion for the next channel number value is made under a uniform distribution assumption applied to the rest of the frequencies of the hopping set.

Memoryless case happens to be less stringent than previous, allowing for the current and the next outcome (*v*) to be the same; and again, a uniform distribution selection criterion is also considered as well. In this case, $P(f_{j+1} = f_j) = q^{-1}$ [25], [27]. As it can be seen from Figure 2.2, the same value of frequency could be generated in two consecutive time slots. However, on the average, it is guaranteed that all the frequencies will be equally used.

In case of deterministic hopping patterns as a general principle we have that given a prime number *q*, it is possible to obtain a set of $N = q - 1$ non-repeating hopping codes or sequences $\left\{ f^k \middle| 1 \le k \le N-1 \right\}$ holding an acceptable cross-correlation property. Each of the sequences f^k from the set has period *N* and is a kind of non-repeating codeword or sequence (i.e., $(f_j^k \ne f_{j+n}^k \middle| 1 \le n \le N-1)$). In order for such a set to hold an excellent cross-correlation property, equation (2.2) must hold for any pair of given codewords f^i, f^k and for all values of *j*:

$$\sum_{n=0}^{N-1} \delta(f_n^k, f_j^i) \le 1 \tag{2.2}$$

Where the function $\delta(\cdot, \cdot)$ is defined as $\delta(u,v) = \begin{cases} 1 & if\ u = v \\ 0 & otherwise \end{cases}$.

One of the advantages of Spread Spectrum technique is that it allows for CDMA operation. This means that several co-located systems as part of a wireless network could transmit using different hopping patterns governed by special designed pseudorandom sequences. Hopping frequencies that correspond to such hopping sets are usually the same, but they could be generated in a specific way such that rarely two or more systems

(active users) will collide (i.e., transmitting at the same RF frequency at the same time). This is true when a set of codes holds a very low correlation value for any given pair of codewords from the hopping set or family.

In a multiple access environment, it is always desirable that interference derived from this principle of communications be as low as possible. Therefore, the cross-correlation property that is intrinsic to a set of codewords or hopping sequences becomes crucial. Markov and Memoryless-based hopping patterns are frequently referred to in low-power ISM applications and within the research literature regarding the topic of our concern here [9], [25-29]. However, code sets based on these random general stationary processes are precisely not known for holding good behaviour in this sense. What is commonly guaranteed besides the way a codeword is built is the statistical independence between any pair of them within a set.

It is possible to generate a number of codes comprising a hopping set in order to deal more efficiently with a multiple access environment as stated before. It is also desired that within our application context, there is a certain level of resilience to code mismatching due to shift that is induced in the time domain. In other words, the number of coincidences between any pair of codes from a set should be kept low in order to cope for clock imperfections. This will imply that even when clock drift causes relative code shift in time domain (hopping patterns will certainly become unaligned with respect to each other), the above-mentioned property still needs to be attractive from the system performance point of view.

Frequency-Hop codes based upon the theory of Cubic Congruences (CC) have its foundation on the theory of numbers and congruences. In this sense, it works around algebraic structures. This is something appropriated according with the FHSS approach commonly used in low-power ISM SoC applications (i.e., look-up table). An in-depth development of the number theory applied to obtain good correlation signals could be found in [24] and a concise refresh in congruence theory could be found in [30].

CC codes are obtained as ordered sequences of integers, through the placement operator $y(k)$ over what is considered a finite field of order *p*, denoted here as $J(p)$, and defined with respect to the prime number *p* as follows [31]:

$$y(k) \equiv ak^3 \,(\mathrm{mod}\, p) \tag{2.3}$$

Where $(\equiv)$ implies congruency, $k \in \{0,1,...,p-1\}$ and $a \in \{1,2,..,p-1\}$. The last parameter will define a unique member of the code set or *class* (i.e., a specific sequence or codeword), since $y(k)$ defines a class of cubic congruence operators or a complete residue system with respect to the prime number *p*. Obviously, the parameter *a* will also define the number of codes in a set. For instance, if $p = 41$ then only a number of 40 low mutually-correlated codes are possible to be attained by applying equation (2.3).

The totality of integers generated by the product ak^3 in the previous equation is congruent modulo p with $y(k)$. The placement operator $y(k)$ can be seen as a binary operation over the set $J(p)$ that maps ordered pairs in the form of (a_i, k_j) in the set $J(p)$. This placement operator is also a *permutation* of the set $J(p)$ (i.e., $y(k) \in J(p)$) as can be seen in Figure 2.3. Hence, there will be a one-to-one correspondence between the above-mentioned pairs and the corresponding image $y(k)$.

$$y = \begin{pmatrix} 0 & 1 & 8 & 5 & 9 & 4 & 7 & 2 & 6 & 3 & 10 \\ 0 & 2 & 5 & 10 & 7 & 8 & 3 & 4 & 1 & 6 & 9 \\ 0 & 3 & 2 & 4 & 5 & 1 & 10 & 6 & 7 & 9 & 8 \\ 0 & 4 & 10 & 9 & 3 & 5 & 6 & 8 & 2 & 1 & 7 \\ 0 & 5 & 7 & 3 & 1 & 9 & 2 & 10 & 8 & 4 & 6 \\ 0 & 6 & 4 & 8 & 10 & 2 & 9 & 1 & 3 & 7 & 5 \\ 0 & 7 & 1 & 2 & 8 & 6 & 5 & 3 & 9 & 10 & 4 \\ 0 & 8 & 9 & 7 & 6 & 10 & 1 & 5 & 4 & 2 & 3 \\ 0 & 9 & 6 & 1 & 4 & 3 & 8 & 7 & 10 & 5 & 2 \\ 0 & 10 & 3 & 6 & 2 & 7 & 4 & 9 & 5 & 8 & 1 \end{pmatrix}$$

Figure 2.3 Set of codewords (Cubic codes of order 10) for *p*=11. *J (p)* = {0, 1, 2, …, 10}

Generally being the set $J(p)$ defined with respect to the prime p as $J(p) = \{0,1,...,p-1\}$, the most complete definition for such a field is by naming it as a Galois Field (finite field) of order p ($GF(p)$), for which the following equivalence holds:

$$J(p) \equiv (Z_p^+, +, \cdot) \equiv GF(p) \tag{2.4}$$

Where Z_p^+ in this case denotes a set of positive integers modulo p over which two binary operations called addition and multiplication are defined.

Obviously, the fact that p needs to be an odd prime number limits the generality of the method; however, it is not that critical as it is shown next. There is also another constraint regarding the prime number p and relates the capacity of the method to produce a family of *full frequency hop codes*. As can be seen in the set of codes presented as example, each of the codewords (rows in the matrix) belonging to a given set fully expands over all the members of the finite field $J(p)$. In order for this to happen, the prime number p has to satisfy the following equality [31]:

$$p = 3m + 2 \tag{2.5}$$

Where *m* should be a positive integer. This restricts the method by allowing for about 50% of the existing prime numbers for its development. Examples of prime numbers that satisfy equation (2.5) and for which the aforementioned capacity holds are as follows: 5, 11, 17, 23, 29, 37, 41, 47, 53, 59, 71, 83, 89 and 101. These values are very close to the typical choices for the number of hopping carriers in real-life systems (considering range from 5 to 101 channels) and in this case constitute about 60% of all the prime numbers in that range.

Markov and Memoryless code sets do not provide for a *full hopping code* most of the time because of their formation law. This property should be understood as the capacity of a codeword of containing all the values of the finite set over it is conceived. Such a property is fully guaranteed by CC [31]. It is noteworthy to point out that the placement operator in equation (2.3) ensures for this characteristic because of the way the prime number *p* is chosen.

Regarding the resilience exhibited by this code to the number of coincidences as it is shifted in time domain, it is shown in [31] that at most three collisions (i.e., the number of components between any given two codewords that match) will happen for the same prime number *p*. This is something theoretically supported by the fact that the *placement operator difference function* given by equation (2.6) has a maximum of three non-congruent roots for *k* (i.e., by making the previous equation equal to zero and applying the Lagrange's theorem as stated in equation (2.7)). This function is evaluated based on the difference between any two placement operator values generically considered as y_1 and y_2 [31]:

$$(y_2 \Delta y_1)(k;t,w) = y_2(k+t) + w - y_1(k) (\mathrm{mod}\, p) \tag{2.6}$$

$$(y_2 \Delta y_1)(k;t,w) \equiv 0 (\mathrm{mod}\, p) \Rightarrow (a(k+t))^3 - (bk)^3 \equiv 0 (\mathrm{mod}\, p) \tag{2.7}$$

Where *t and w* refer to time and frequency domain respectively. In this sense and more generalized, a time-frequency shift is considered. However, in our case, we would just be interested in any displacement in time domain. In equation (2.7), the parameter (*b*) plays the same role as parameter (*a*) in equation (2.3).

Related with the previous idea, an upper bound (usually claimed to be uniform over the entire code class and range of code displacement) for the maximum number of collisions is possible to be attained in case of the CC family of codes [31]. This fact is normally verified at the simplest level (i.e., by considering just any two codes from the whole set). In the context of our experiments, this fact was verified by the uniformity observed with respect to the maximum number of collisions that the code set exhibits when successive shifts are applied to it in the time domain.

Based on what was introduced in Section 2.1 regarding the practical way in which FHSS systems are currently implemented in low-power ISM transceiver applications, a unique look-up table could be assigned to each of the master-slave association in the network.

The feasibility of implementation for this family of hopping codes in regards to our application context is evident.

2.3 System Main Performance Metric

In [11] and [32] the performance of a *single* real – time FHSS packet-based RC system that operates in the ISM band was completely characterized by means of the *SLOP*. No other previous research work has addressed the concept of probability of a lag occurrence in order to quantify the user experience during a typical radio control session.

In presence of real-time RC applications based on packet radio, a lag is verified whenever the system latency exceeds the Human Response Time (*HRT*), assumed as 100 ms [11]; this is something obvious since at any given time instant during a real-time radio control session, the stimulus applied to the system is simply an input from a user.

In case of FHSS systems, *SLOP* can be related to the synchronization process as a whole. To this effect, the synchronization algorithm referred in [11] known as the *modified transmitted reference* (see Figure 2.4) was found very convenient. In fact, the algorithm results quite suitable for packet-based slow FHSS systems [17]. It is based on the assumption of a serial search acquisition engine enabled by a matched filter. In this sense, for every packet sent by the transmitter, the matched filter sub-system will search for a preamble or special synchronization sequence. This is known as the *search state* within the algorithm and once it is correctly detected, the transmitter-receiver pair has reached what it is known as the *acquisition state*. At this point the receiver will initialize its own hopping process based on the long sequence. Since it is known at the intended receiver, it keeps hopping synchronized by tracking the number of received bits once the timing reference obtained from the matched filter has been established.

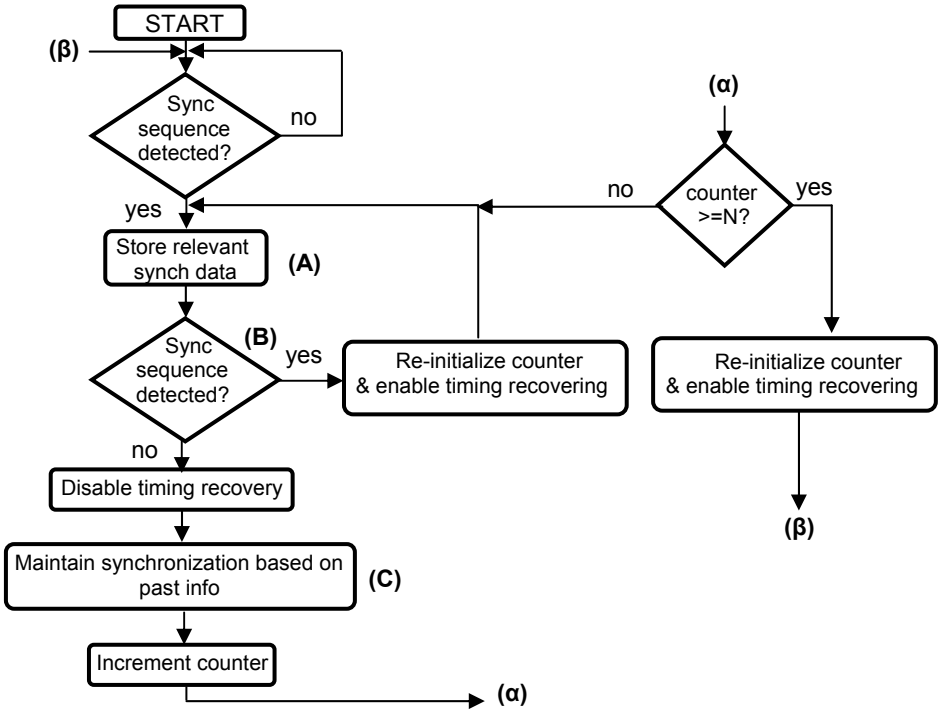


Figure 2.4 Modified transmitted reference synchronization algorithm [17]

The robustness of the modified transmitted reference algorithm relies on the possibility of *maintain synchronization based on the extrapolation of timing data* which is stored at every correctly received packet (labeled as *A* in Figure 2.4). The fact of a packet being

received correctly will represent a new possibility for the system to keep the tracking condition normally for longer. Since information about the past states of the long hopping sequence, the number of received bits within the current packet, and the length of it is of knowledge for the receiver, it can be able to maintain acquisition lock (labeled as *C*) in case the time synchronization sequence was not detected (labeled as *B*). Based on that information, it will be possible for the receiver to correctly commute of frequency at the end of the current packet or time slot.

If the special preamble is not detected properly due to the presence of strong interference at a given hop time, the system assumes that in the next hop channel conditions will differ from the current one and will remain in tracking condition using the last good timing reference instead as explained above. If bad channel conditions persist such that *N* successive packets are not correctly received, the algorithm will force the receiver to go under re-synchronization in order to get a fresh time reference, as shown in the flowchart.

In general, the acquisition phase by means of the initial search process could be time consuming. Being under in-lock condition, a FHSS system would only go under search stage due to the effect of independent factors such as channel noise and Multiple Access Interference (MAI) that may lead the system to lose synchronism completely. Such a transition is always undesirable since chances for a lag to occur are in general more likely. While trying to acquire synchronism if the *inter-arrival time* of two consecutive non-corrupted packets exceeds the *HRT*, a lag will be verified. Consequently, system response will not hold the *real-time* condition since control commands that were sent to the remote unit did not reach the receiver at the intended time instant.

A simple inspection of the **two distinguishable** cases when defining *SLOP* within the context of the referred synchronization algorithm will help to draw some interesting facts regarding the adequacy of the metric.

Case 1 considers the time it takes for the synchronization algorithm to declare out-of-lock condition (T_{LS}) is greater than *HRT*. This case is only associated with the stage prior

to re-acquisition; in this case, a lag would happen before the system decides to re-start the synchronization process. Assuming both, statistical independence between data packets and one packet transmitted per hop, it follows that [11]:

$$SLOP = PER^{HRT/T_{dwell}} \tag{2.8}$$

Where *PER* is the probability of a packet to be corrupted and T_{dwell} is the system dwell time. In [11] *PER* is normally associated with the Bit Error Rate (*BER*) of the channel under the assumption of a uniform error distribution hypothesis [33]:

$$PER = 1 - (1 - BER)^{Lp} \tag{2.9}$$

L_p is the actual packet length, usually considered fixed within the application context of our concern. It is clear that when T_{dwell} is decreased in equation (2.8) (for instance, by increasing the data transmission rate while keeping L_p constant) chances for a lag to happen are minimized. Related with this and based on the algorithm in Figure 2.4, considering *HRT* of around 100 ms, it will be less likely that *N* successive erroneous packets will be received. In this sense, the probability for a lag to happen would be lower.

In **case 2**, the opposite happens, as the time it takes for the receiver synchronization algorithm to declare it has lost synchronism T_{LS} is less than *HRT*. A more complex situation given by the interaction of two processes (i.e., acquisition and tracking stages) at the receiver will define the occurrence of a lag. In this particular case *SLOP* is given then by the joint probability of two statistically independent assumed events [11]:

$$SLOP = P_{LS} P(T > (HRT - T_{LS})) \tag{2.10}$$

Where P_{LS} is the probability for the system to go under re-acquisition stage; this directly addresses one of the events mentioned previously. This fact can schematically be seen in

the Finite State Machine (FSM) model diagram shown in Figure 2.5 that could be associated with the algorithm shown in Figure 2.4 and where the uppermost state (designated as *ACQ*) represents the process of re-acquiring synchronism. With respect to this particular state, it is considered as the starting point from where the synchronization algorithm will always initiate. Whenever the receiver system loses synchronization completely or is powered on (indicated by the START condition in the FSM diagram), it will try to acquire time synchronization with the intended transmitter. Whichever of these two events happen, the receiver's hopping sequence phase is shifted by algorithm to a value that corresponds to the middle of the hopping band [11]. When communication starts or during the acquisition processes the transmitter and receiver may be in different RF channels while trying to acquire synch with each other. Due to this, any transmitted packet is considered as *missed* by the receiver regardless its status as it traverses the medium [11].

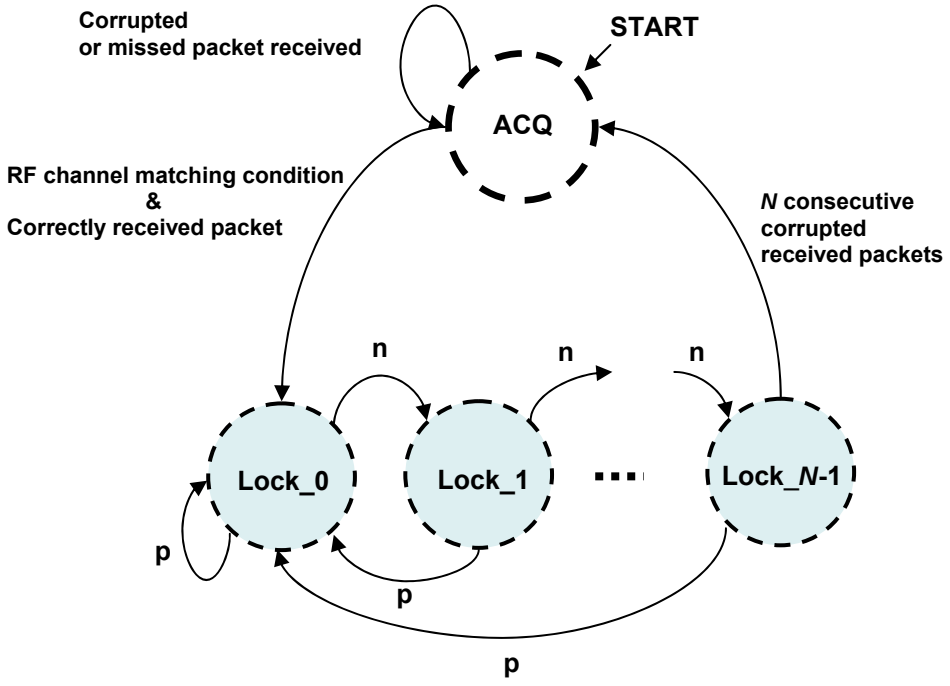


Figure 2.5 FSM diagram associated with the modified transmitted reference synchronization algorithm for the receiver system

In the previous state diagram, the transition probabilities n and p are such that $n = PER$ and $p = 1 - n$, respectively. States from **1** to ***N-1*** imply that a bad packet has been received but the receiver system is still in synch condition. With respect to those states,

whichever be the current state of the receiver if the synch preamble is declared erroneous, the algorithm causes the system status to move forward in the chain with probability *n*. If the opposite happens, the system will go back (if system was at a state other than ***Lock_0***) to the leftmost state, represented as ***Lock_0*** with probability *p*. This state represents the desired condition where the receiver is properly interchanging data packets with the transmitter without forcing the synch condition by means of extrapolating time data from previous successful receptions. Assuming statistical independence between consecutive erroneous packets, it is possible to relate P_{LS} with *PER* as follows [11]:

$$P_{LS} = PER^{N} \tag{2.11}$$

In order for the receiver system to declare out-of-lock condition, *N* consecutive erroneous packets should have been received. In such a case, system status will be shifted to the state *ACQ* and will remain in such state trying to acquire synchronism afresh upon the successful detection of the short synchronizing sequence performed by the matched filter subsystem. From equation (2.11), as the value of the parameter *N* is increased as part of the synchronization algorithm, there would be fewer chances for the receiver to go under re-acquisition for the same channel conditions, something that could also be inferred from the FSM diagram. This will certainly decrease the likelihood for a lag to occur.

However, in order for a lag to verify as stated before, another event must occur given by the probability: $P(T > (HRT - T_{LS}))$. This event verifies whenever the system is not able to acquire before the remaining time given by the difference between the *HRT* and the time it takes for the system to declare itself out of lock (T_{LS}) elapses. This event has to deal entirely with the acquisition process where the search strategy used will play an important role in this probability. If for instance, a uniform serial search algorithm is employed, each of the channels in the hopping band is serially scanned in order to find which of them satisfies the matching condition with respect to the transmitter (i.e., same RF channel). In this case, only the level of affection due to the channel impairments will define the system performance.

Despite what has been mentioned above regarding [11] and [32], the fact of considering a plurality of users scenario by modeling a set of Frequency-Hop (FH) codes o patterns was not implemented. The impact of MAI on the performance of an individual system in a wireless multi-user environment is well-known. Given a total of *K* active users in the network, the average probability of symbol error at any given user provided that there are *K-1* interfering users, will depend on the kind of FH pattern employed [25]. This is something that is examined in this research work. In concordance with what can be found in the related literature, both random general stationary process and a type of deterministic based FH pattern sets were considered in this work with the aim of evaluating system performance in a more realistic scenario. Finally, the fact of modeling multiple users also facilitated a way to impose and evaluate certain technical constrains of interest for this research, such as variable packet duty cycle and imperfect clock oscillators governing the hopping sequence at every active user.

2.4 Channel and Transmitter-Receiver Models

Models developed in [11] for the channel impairments (i.e., interference sources and noise) and the receiver system reflect the reality at a certain satisfactory level and will extensively be used as a modeling reference throughout the present research. They are closely related to each other as the way the receiver evolves in time will highly depend on the status of the channel at every hopping period.

The channel model in [11] was conceived such that two channel types (or status) are possible: *Blocked* channels considered under 100% of *PER* (presence of strong co-channel interferers) and the *non-blocked* channels associated with a certain *PER* due to other sources of interference not specified in [11]. The latter is modeled by means of a random generator which is implemented for each channel of the hopping band. The generator's output will dictate the status of the channel at every hopping period according to the indicated rate (i.e., defined value of *PER*).

The FHSS transmitter model is simply a subroutine that cyclically runs on a specific ordered set or a list of integer numbers (hopping sequence) representing the allowed

hopping channels. The slow FHSS receiver system operates under the well-known uniform serial acquisition scheme enabled by a matched filter. The modified transmitted reference synchronization algorithm is assumed as the theoretical base for the model as it is known for its suitability for packet-based FH radio [17]. A FSM model suggested for the receiver properly describes the states transitions according to the aforementioned synchronization algorithm. More details about the operation of the receiver algorithm can be found in Section 2.3.

Chapter 3

Model Implementation

The present chapter comprises four sections that provide a detailed explanation of what has been specifically developed in this research work regarding the model framework. Particularities associated with each of the scenarios mentioned within point three of our research objectives are analyzed. As stated in the previous chapter, due to its suitability, we basically followed the core of the modeling developed in [11]. However, new aspects have been introduced, for instance, when conceiving the multi-user environment and the interference model.

Section 3.1 addresses certain new aspects that are considered in the interference phenomenon when conceiving a more complete and realistic channel model. Based on what is treated in the previous sections, it follows Sections 3.2 and 3.3 where details about the modeling of a SFHSS-MA network operated with ideal and real clock oscillators are provided, respectively. Finally, in Section 3.4, the AFHSS-MA scenario is analyzed as the more generalized multi-user environment.

3.1 Channel Impairments Modeling Issues

No co-channel interference, specifically due to MA, is modeled as a set of a predefined value of 100% of time affected channels as it was done in [11]. Since we deal with a multi-user scenario where a specific set of hopping codes is modeled, this kind of interference is essentially left to the sake of collisions happening at every dwell time. This interference obeys a probability of occurrence that increases as the number of users accessing the medium does. This important issue will be treated more in depth in Section 3.2.1.

On the other hand, due to its nature, RF emission is vulnerable to interference from other sources. This has become a real and serious problem for commodity technologies that share the ISM band [34], [35]. Interference phenomenon due to external sources has been

considered here specifically as partial-band interference, as it has been pointed out as the most harmful effect to slow FHSS systems.

In order to be as realistic as possible, the presence of interference sources that are exogenous to the system should be taken in consideration [35]. They are, among others: commercial 2.4 GHz cordless phone systems; Bluetooth personal area devices; microwave ovens (with 50 percent of duty cycle which create a jamming pulse in the above-mentioned band); and low energy RF lighting sources. Interference could potentially occupy a portion of the entire hopping frequency spectrum (either as concentrated over a given spectral region or through some isolated frequencies or bands). The nature of this interference can be modeled as partial-band interference; being in this case, the "jamming" signal modeled as a zero mean wide-sense-stationary Gaussian noise process [12], [18]. Based on this, it will exhibit a flat power spectral density over a fraction (α) of the entire hopping bandwidth (W_{ss}). Following this line of reasoning, it could be possible to think of the partial-band interference distribution at any time instant to be comprised of narrow band jammed portions that are spread out all over the hopping band W_{ss}. This fact can be related to a probability that corresponds to the aforementioned percent (α) that there will be active users whose RF emissions will be affected at every dwell time [12], [18].

How fast the interference distribution pattern changes (i.e., the distribution of the set of channels that are under interference effects) is something that has been assumed here in the order of a dwell time period. This means that we assume a constant interference pattern during at least that amount time. It is something that we believe strongly depends on the dynamic of the environment where radio communications take place. For instance, an urban and a rural-like scenario will not behave the same from the interference perspective (in how aggressive and dynamic it could be).

Depending on its intensity, the interfering signal represented by J_o in equation (3.1) could make the Signal-to-Noise-plus Interference Ratio (*SNIR*) low enough as to make a

channel at a given time instant (i.e., at dwell time) completely useless. The *SNIR* is assumed as follows:

$$SNIR = \frac{E_b}{N_o + J_o} \tag{3.1}$$

Where $E_b = \frac{P_{av}}{R}$ is the bit energy in [watts x sec], P_{av} the useful signal average power, R is the information rate in [bits/sec], N_o is the thermal noise power spectral density and $J_o = \frac{J_{av}}{W_{ss}}$ is the interfering signal power spectral density both in [watts/Hz]. In this last expression, the numerator is the average received power of the interfering signal [12], [18].

In our model, as in [11], each channel status will dynamically move between two states depending on the level of interfering signal that is present during the dwell time. For instance, a channel is considered as "good" when its *BER* is low enough and will only depend on the well-known Signal-to-Noise Ratio (*SNR*) [18], [36], and [37]. In this desirable case, the quality of the channel is just associated to the Additive White Gaussian Noise (AWGN) behaviour. On the other hand, when the interference level is much higher than the thermal noise floor, the term J_o in equation (3.1) is much greater than N_o and the channel is then considered under strong interference. As a consequence, the received packet is declared corrupted. The reason that it may lead to this fact could be either the level of interference due to the presence of co-channel or in-band RF emissions, which correspond to same system and external-to-system interference respectively.

Following what has been considered above, every channel of the hopping band has been modeled such that there is a random generator associated with it [11], [36]. Two states (i.e., good or bad) are possible for the channel as we stated before; the generator corrupts

the associated channel at the pre-selected probability *PER* (see Figure 3.1), which in turn is closely related to the parameter (α) introduced earlier in this section. Channels are assumed to behave statistically independent with respect to each other through the whole simulation session.

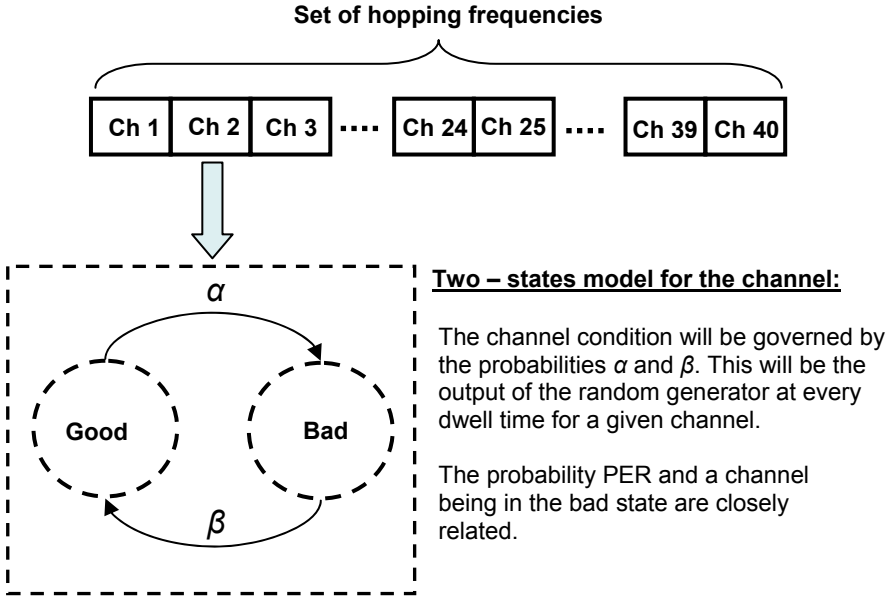


Figure 3.1 Model for the channel as interference is considered

Related with Figure 3.1, Table 3.1 shows the possible states for the noise generator output and the value assigned to the status of the channel at every dwell time.

Table 3.1 Mapping of generator output-channel status

Output of the random generator at channel *ith*	**State of the channel ith**
0	Good
1	Bad

Since we are dealing with a multi-user environment, at every dwell time the random generator is invoked in order to check for the channel condition in which each of the active users is transmitting. Figure 3.2 shows the kernel of the analysis that is performed on each packet as it traverses the channel and finally gets to the receiver. Initially, before a packet is sent, its status is set to non-corrupted. Recall that a single packet is sent at

each system hop. Hence, the status of a packet is initialized as "good" at every hop period. Generated packets are assumed, as in [11], to be mutually statistical independent.

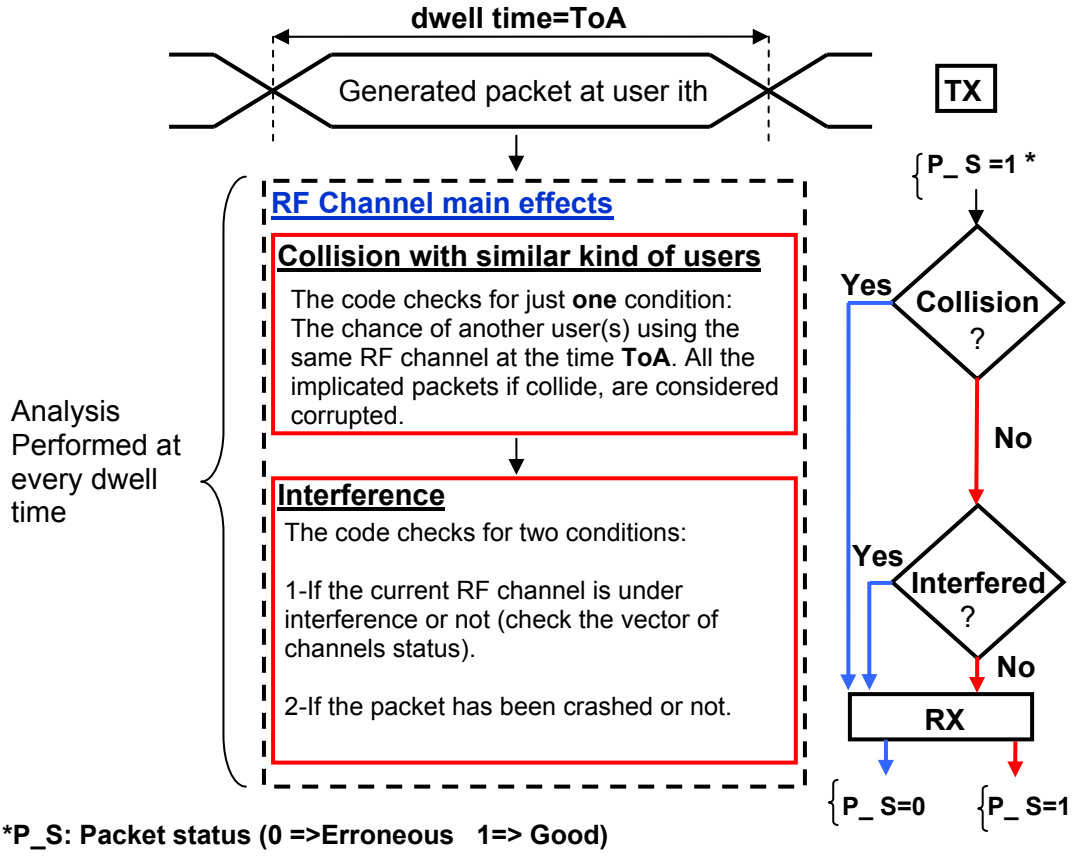


Figure 3.2 Analysis performed on a packet as it is generated at the transmitter (TX) and reaches the receiver (RX). The blue and red paths are complementary to one another's implications

At every dwell time and after the analysis presented in the aforementioned figure is performed, the following updates take place per user basis:

1. Number of correctly-received packets;
2. Number of corrupted packets;
3. Number of missed packets;
4. Inter-arrival time whenever the status of the packet is good;
5. Lag event, if it happens (based on previous point), is recorded.

3.2 Synchronous Frequency-Hop Spread Spectrum Multiple-Access (SFHSS-MA) Scenario with Ideal Clock

A FHSS-MA network can be categorized into two schemes according to the grade of synchronism between users hopping patterns. As this case suggests that, theoretically, there will be a perfect alignment between them in time domain as shown in Figure 3.3 [28].

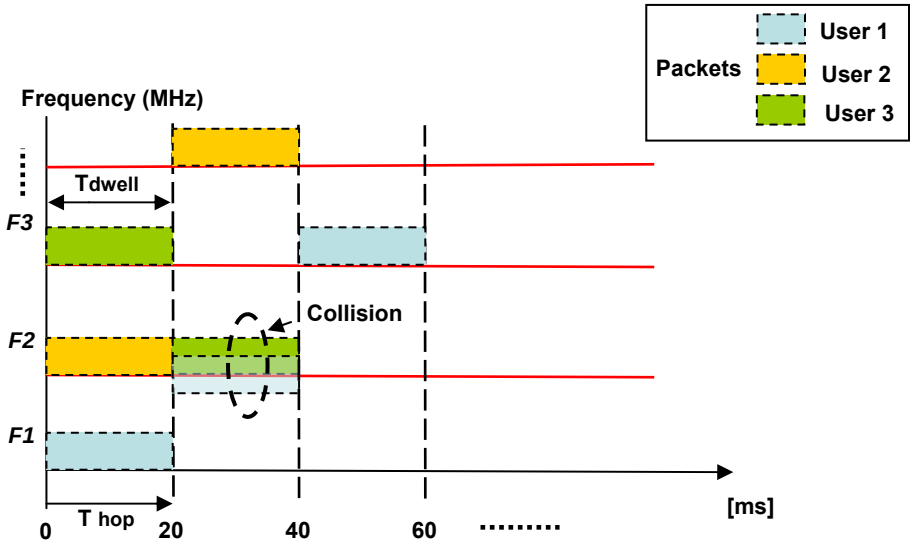


Figure 3.3 Synchronous FHSS-MA scenario with ideal clocks

The kind of FHSS operating scheme simulated here, as stated earlier, is the well-known SR in which a pseudorandom code is known at both the transmitter and the receiver sides.

As can be seen in the diagram, all the users join the network for the first time at time instant $t = 0$ ms, adopted as a temporal reference here. From this point in time, users will keep hopping with a fixed hop period of 20 ms, which also corresponds to a clock tick. Based on this, the *dwell time* or equivalent to the *ToA* is considered to be 20 ms. In what follows within this scenario, we may interchange between these two terms and *time slot*, as indicated in the previous mentioned figure, since all of them refer to the same magnitude.

Regarding the way a plurarity of users (i.e., a network of real-time RC FHSS applications) is implemented, we have that the bottom line for the modeling of this aspect has been the implementation of a set of statistically independent sequences or hopping patterns via pseudorandom generators in a computer. The output at each random generator is supposed to follow a certain statistical principle. Normally, we build a set of hopping patterns (i.e., an array of codewords) according to the desired number of RC applications to be tested, and then they are periodically repeated as the simulated radio session lasts.

In conclusion, the modeled network could be thought of at this point as several master-slave associations interchanging packets at a period of 20 ms perfectly synchronized (no clock drift is considered) with respect to each other. A specific correlation property for the set of hopping sequences is not considered at this stage. Related with this, we expect a significant number of collisions per time slot. As is shown in Figure 3.3, if two or more users jump to the same frequency they will collide 100 percent of the time. An example of this situation would be the collision of packets sent by users **1** and **3** at frequency *F2* within the time slot from 20 to 40 ms that can be seen in the same figure. Besides the channel modeling issues explained in Section 3.1, intra-network collision causes interference as well (i.e., self-system interference) and they together will impact the system performance. Of course, it is something that could be alleviated in this kind of scenario by implementing a hopping set which exhibits an attractive cross-correlation property. We will touch this feature more in depth as part of a more realistic SFHSS-MA scenario later in this work.

3.2.1 Collision Analysis

In general, errors in multiple access systems are due primarily to MAI. In considering a collision or hit at the hopping level, we have adopted the idea given in [25]: a hit at user i^{th} from the k^{th} user is verified if $f_k(t-\tau_k) = f_i(t)$ for at least one value of *t* within the l^{th} hop interval $\left[lT_{dwell}, (l+1)T_{dwell}\right]$. Parameter τ_k is any delay (if considered) at the hopping pattern level. This means in the wide sense that if the phases of any given pair of

codewords or hopping sequences (f_i, f_k) that belongs to a network are identical (something that is extensively applied to an asynchronous FHSS scenario), a collision of packets that are being transmitted by two or more active users will verify. This fact will become an additional source of system performance degradation.

When considering a multiple user scenario with *K* FHSS active systems, the probability of one or more collisions from the $K-1$ network users at user i^{th} during one dwell time in the case of mutually independent random general stationary processes is given by [25]:

$$\hat{P} = 1 - (1 - P_h)^{K-1} \tag{3.2}$$

Where the term P_h refers to the probability of user i^{th} is hit by at least one user (user k^{th}) in the network in presence of AWGN or a nonselective fading. For Memoryless type FH patterns, this probability is given by [25]:

$$P_h = q^{-1}\left(1 + \frac{1}{N_b}\left(1 - q^{-1}\right)\right) \tag{3.3}$$

N_b is the number of data bits transmitted in one dwell time. In case of Markov based FH patterns, we have for the same conditions, that [25]:

$$P_h = q^{-1}\left(1 + 1/N_b\right) \tag{3.4}$$

This latter probability is in general greater than the one given by equation (3.3). However, when the number of hopping channels (*q*) is increased, they behave very similarly. For example, when considering values for q and N_b of 40 distinct RF channels and 160 bits respectively, we obtained the same probability $(P_h = 0.0258)$ for both cases.

As an illustrative example, the dependence of the probability given by equation (3.2) evaluated for the Markov case, with respect to the number of distinct RF carriers and the number of active users in the network could be seen in Figures 3.4 and 3.5, respectively.

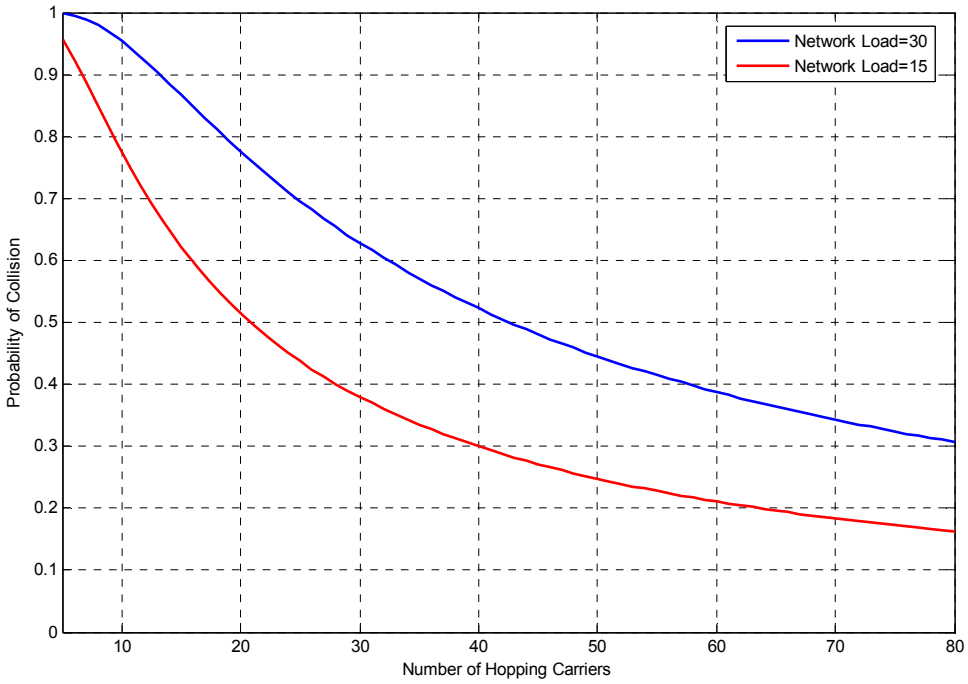


Figure 3.4 Dependency of probability of one user being hit by the rest of the active users in the network as the number of RF channels is changed (theoretical)

It can be noticed that as the availability of distinct RF carriers for hopping increases, the chances for collision decreases, something that supports what was stated as a conclusion in [11]. Since fewer collisions are expected, the inter-arrival time of correct packets decreases and this in turn increases the performance of our system eventually.

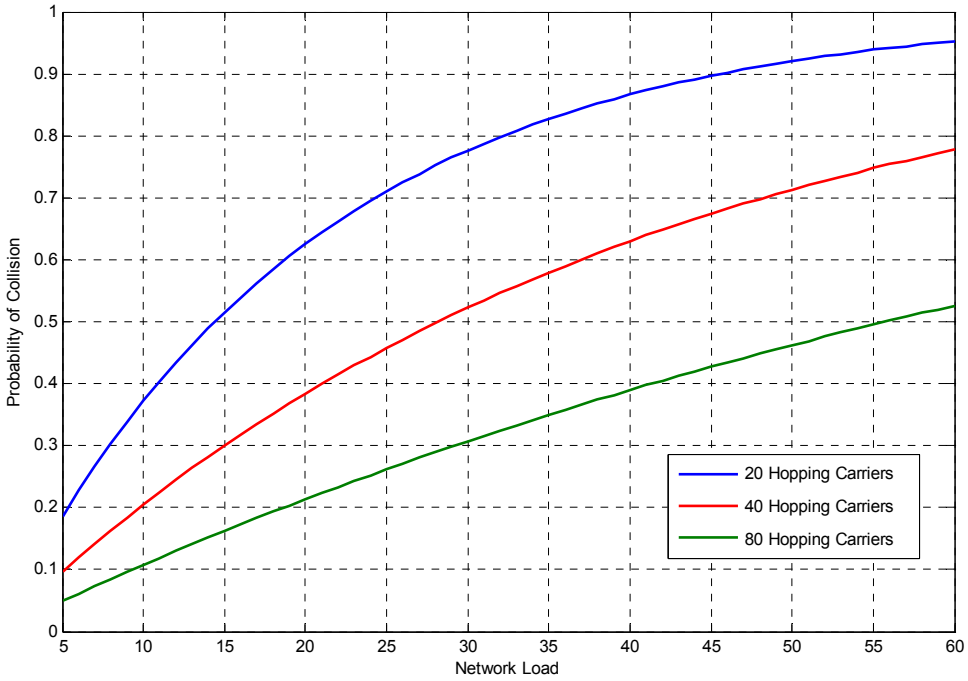


Figure 3.5 Dependency of probability of one user being hit by the rest of the active users in the network as the network load is varied (theoretical)

It is also possible to note that as the number of active users is increased, the performance referred to a single user/application gets worse, something that was verified in our simulated scenarios. By intuition, as the number of possibilities for a given channel to be simultaneously used increases, more chances for collisions are to be expected for a same number of RF hopping frequencies. It was found that system performance was more sensitive to an increasing in the number of users in case of a low number of RF carriers (comparing curves for 20 and 80 RF channels for an increase in the network load from 15 to 20 users in the previous mentioned figure).

In case of deterministic FH patterns, the probability $\hat{P}$ will differ from the general form given by equation (3.2) being in this case [25]:

$$\hat{P} = 1 - \left\{ \prod_{\substack{k=1 \\ k \neq 1}}^{K} \left[1 - P_h \right] \right\} \tag{3.5}$$

In this case, an upper bound is set for P_h as different from previous cases [25]:

$$P_h \leq P_h^{'} \stackrel{\Delta}{=} \frac{1}{1-q} \left(1 + 1/N_b \right) \tag{3.6}$$

Inequality given by equation (3.6) will impose an upper bound on the probability given by equation (3.5), which roughly corresponds with the case for the Memoryless based FH patterns for large values of *q*. The condition imposed by equation (2.2) is very strong and could make equation (3.6) very low or zero as it is the case of the kind of deterministic hopping code set that is discussed in Section 2.2.

As the simulation time evolves, the effects of noise and interference are analyzed together at every dwell time (as shown in Figure 3.2). Besides the status of the interference due to noise at every channel that is used at a given time slot, we analyze how many packets were corrupted due to collision events. It is clear that the only condition to be evaluated under this circumstance is as follows:

$$RFchannel_i = RFchannel_j \tag{3.7}$$

Where $i, j \in \{1,2,...,numbActiveUsers\}$ and $(i < j)$. Equation (3.7) simply evaluates the event that two or more users are using the same channel at a given time slot. If it is satisfied, packets sent by users *i* and *j* are both automatically declared corrupted and counted. This is the main difference between our model and respective analysis with [38] when considering interference other than noise itself, since we are dealing with a multiple access network.

3.3 SFHSS-MA Scenario Implementing Real Clock Oscillator

In the specific scenario we are analyzing here, an initially hopping-level synchronized network is considered. As it was previously explained in our problem statement, each master-slave association will interchange packets with fixed length at a given rate, commonly known as *message rate* and denoted here as (*Tc*) as it is depicted in Figure 1.2 [39]:

$$\text{Channel Period} = \frac{\text{Reference Clock (in Hz)}}{\text{Message Rate (in Hz)}} \tag{3.8}$$

The denominator in equation (3.8) could be thought of as the number of messages per second sent over the channel, and it could be typically found as an adjustable parameter in a specific range (e.g., from 0.5 to 200 Hz), while keeping the packet payload fixed to a certain size value (8 bytes, for instance). The value in equation (3.8) is normally associated with an integer value that is stored in the transceiver memory (by means of an internal configuration register). Also from the same expression, the numerator is commonly referred as to the output of a reference oscillator (clock) driven by a quartz crystal. This reference value is commonly implemented via an external oscillator or could be internally synthesized from an external reference, as is shown in Figure 3.6.

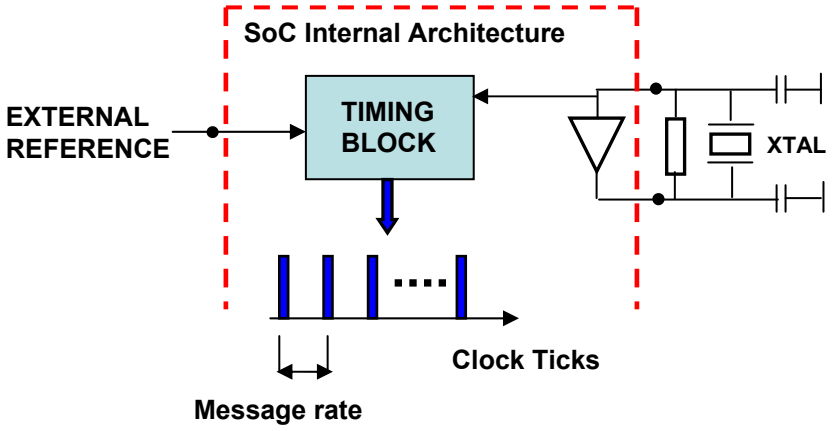


Figure 3.6 Typical timing system seen in most ISM low-power SoC

For instance, the quartz-based oscillator model CO-402A-OX from Vectron Inc. has the main electrical parameters as follows [40]:

Table 3.2 Typical specifications for a quartz-based clock

Output	10.24 MHz
Output Level	TTL compatible (up to 10 loads)
Input voltage	5V dc ± 5%
Accuracy	± 0.005%
Temperature Stability	0.0025% =25 ppm/(0ºC - 70º C)

It happens that at any given time the output oscillator's frequency will differ from the desired specified frequency or named plate frequency; in the example above, it would be 10.24 MHz, resulting in a frequency error. Usually this error is comprised of three primary factors: Initial accuracy, temperature stability, and aging [40]. While Vectron, for instance, keeps separate values for the initial accuracy and temperature stability, these factors may be combined in an overall allowable error with no frequency tuning adjustment. The appropriate term is *frequency – temperature accuracy* or simply *tolerance*, and it is the maximum allowable deviation from the specified nominal frequency, again over a temperature range [40], [41].

The tolerance factor will impact the system performance in different ways. For instance, within the transceiver RF synthesizing section, care must be taken at the time of choosing a clock (i.e., its tolerance value). To this effect, system allowable bandwidth or channel spacing will lead the selection criteria [39], [41]. Also, clock imperfections (which manifest itself as a deviation with respect to the initial clock's frequency) could compromise the nature of the correlation property that a given hopping set holds at a certain point, by continuous variation induced in the message rate in equation (3.8). In doing so, we have added an extra complexity to the model, being this scenario even closer to the reality with respect to the previous FHSS-MA case.

The differences in the period between any two given clocks (i.e., belonging to any given pair of users) could be very small. For instance, in our case the targeted tolerance is:

$$F_o = F_{nom} \pm 1.5\,ppm \tag{3.9}$$

Where F_o is the real frequency value at the oscillator's output and $F_{nom} = 1/T_{nom} = 1/20ms = 50Hz$. However, the accumulative difference over a large number of oscillations could be noticeable enough based on the nominal value for the output frequency. This is an important point to be tracked carefully in this specific section, as it will give an idea on how degraded the system performance can be as time elapses.

For the model implementation concerning this scenario, we conceived a situation based on what was shown in Figure 3.3, which corresponds to all the hopping patterns perfectly aligned at the time instant equal to zero. Under this idealized case (i.e., perfect clock), hopping patterns would evolve completely aligned in time. Figure 3.7 shows a modified and more realistic version with respect to what happens in the aforementioned case, where the relative shift of the hopping codes induced by imperfect clocks is taken into account.

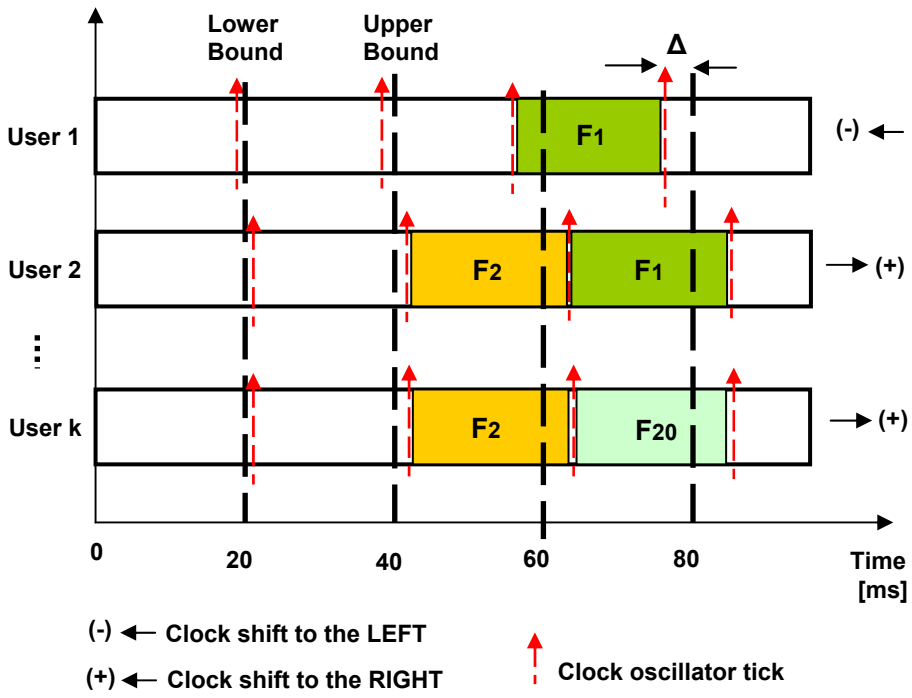


Figure 3.7 SFHSS with imperfect clock. Values (*F*) represent current RF channels that are used to transmit the packet

Based on the previous mentioned figure, some aspects from the modeling perspective are described as follows. For instance, the way each user's hopping sequence evolves in time is represented by horizontal bars vertically split by vertical parallel dashed lines that delimit the duration of a packet transmission (i.e., dwell time). These intervals of 20 ms each would directly be associated to the dwell time in a scenario where ideal clocks are employed. In this new scenario, they are retaken, but only as time references since the real value of *ToA* would not be 20 ms as real clocks are employed. As a time reference, these intervals are used to delimit what we call here *lower* and *upper* bounds and they are normally associated with each of these time periods (for example, as indicated in the diagram, specifically for the time period ranging from 20 to 40 ms). The role of these bounds will change as time passes by. For instance, a time reference that was taken as the upper bound in a previous network activity period will be the corresponding lower bound in the very next activity period, and so on. Within this particular scenario, we also use the concept of *network activity period* that is related with the interval of time comprising same order user clock oscillations (factor *n* in equation (3.10)) and the corresponding RF channels used by each of them to transmit a packet. To explain, we have that the first

oscillation of the clock for all the users happens at around 20 ms, as shown in the diagram in the same figure. In some cases (such as *user 1*), the time instant a given user's clock ticks is below 20 ms, and in some others it is above the 20 ms reference (such as *user 1* and *k*). This fact will define more accurately what we have considered by network activity period in this specific context. This concept will help in the *a posteriori* analysis performed through all the active users in the network as simulation time is running.

The main factor that causes system perturbation within this new condition is the usage of real clocking systems. This phenomenon is considered in equation (3.9) and is modeled by defining, as part of our initialization block, the sign of the clock drift for each user's clock in the network in a random manner. If the sign of the clock drift of a given user results to be negative, for instance, then this clock will shift to the left at a constant rate (e.g., *user 1* in Figure 3.7). This rate is obtained from equation (3.9). As can be seen in the diagram shown in the same figure, as the network starts hopping, each user's message period or hopping time will vary according to the specified clock drift. As mentioned earlier, there is an accumulative effect present which is normally related to the clock drift phenomenon and is graphically represented by the continuous increase of the time gap (**Δ**) (given by equation (3.10)). It can be estimated graphically as the time difference between the time instant a given clock is actually ticking and the upper bound of the associated network activity period taken as a time reference.

$$\Delta = n \times Clock\ Drift \ \text{[ms]} \tag{3.10}$$

The *n* factor accounts for the number of oscillations of the clock that have elapsed at any given point in time. When considering our specific clock tolerance already introduced in by means of the equation (3.9), we have that the amount of offset time in seconds that will be added to or subtracted from the clock's nominal period as the system starts hopping will be given by equation (3.11):

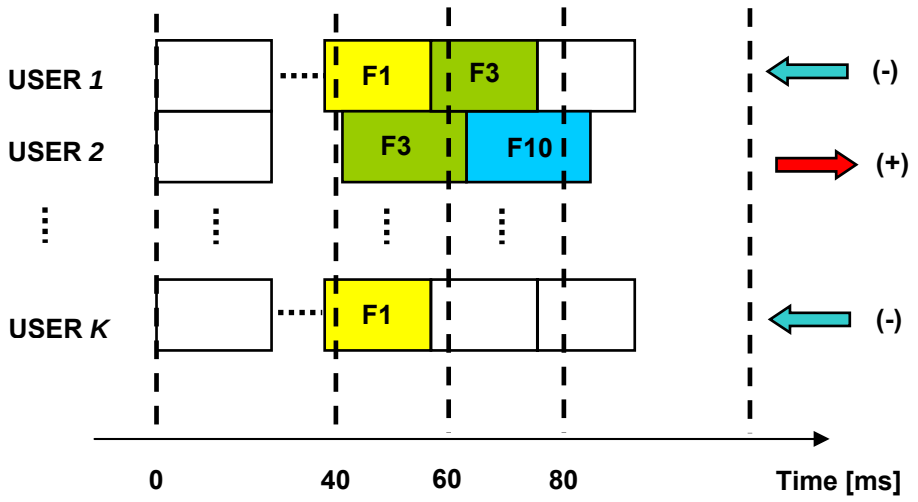
$$Clock\ Drift = 30 \times 10^{-9}\ \text{sec}/\,20\,m\,\text{sec} \tag{3.11}$$

As can be seen, our targeted clock deviates a very small amount at every tick if it is compared with the example given in the Table 3.2. Of course, this is closely related with the quality of the crystal chosen for a specific application.

An important point to be attained in our experiments will be characterizing the impact on system performance of what has been detailed here so far. This is something we believe depends upon a great deal on the design or nature of the hopping codes. Here we recall an important principle: *A FHSS system should be implemented according to a specific application*. As our RC application runs on real time basis, it is important, if possible, to alleviate as much as it can be the effect of interference. MAI, at least, can be attenuated up to certain limits with a robust hopping code design.

3.3.1 Modeling issues: Collision Kernel Analysis, Transmitter-Receiver with Real Clock, and Interference

As master-slave associations start running under the normal FHSS system operation, hopping patterns will start to become unaligned with respect to each other whenever the sign of the clock shift differs (i.e., opposite signs). As explained earlier in this section, this is something that is subject to clock imperfections. This situation will give rise to two kinds of collision events at the hopping pattern level [27], [42]: *full* (something that we have analyzed previously in Section 3.2.1) and a *partial* collision type. A typical scenario detailing this issue is shown in Figure 3.8.



Fxx: Frequency value [Hz]

Figure 3.8 Collision events in SFHSS scenario with real clocks. Partial collision and full collision at frequencies F3 and F1are respectively shown

A complete collision pattern can be defined from the previously referred figure. At this point, we could say that this is something that is not dependable on the nature of the hopping code set. However, the consequences of it are indeed highly dependable on the nature of the hopping pattern.

A *full* collision is verified whenever two or more users make use of the same RF channel to transmit a packet for the same amount of time (i.e., dwell time). For instance, in Figure 3.8 this happens to users **1** and ***K*** when transmitting a packet at frequency F_1 within the activity period between 40 and 60 ms where transmissions of all active users mostly take place. Both packets will be totally hit as the collision lasts the whole dwell time. This event was dominant 100% of the time in the SFHSS case (with ideal clocks) whenever a collision ocurred.

A *partial* type of collision could involve more than two packets simultaneously when analyzing any two given users. In this case, packets that are involved in the collision are affected but only for a given percent of their duty cycle. For instance, this would happen to users **1** and **2** in the above-mentioned figure, within the period ranging from 40 to 80 ms. In this particular case, a maximum of three packets could be involved instead of just

two, as in the previous referred and more trivial situation (i.e., full collision). It is clear that up to three packets could be compromised in a partial collision case when considering just any two codewords from the set at a time (i.e., users **1** and **2** in the example). Of those three packets, one belongs to user **1** which is transmitted at frequency F_3; the other two belong to user **2.** These two packets are transmitted at frequency F_3 and F_{10}, respectively. However, the latter could perfectly be sent at frequency F_3 instead (if Memoryless hopping pattern would be employed). In the scenario taken as a graphical example, we intentionally caused two of them to collide since the same frequency F_3 was used only twice (by users **1** and **2**) in the time interval from 40 to 80 ms. In this case, we say that the packet corresponding to user **1** at F_3 is partially affected by the packet sent by user **2**. From user **1** perspective, this is known as *a collision from the left* with respect to user **2**. The same situation would happen to user **1** (but from the right) if user **2** used the same frequency F_3 instead of F_{10}. However, in this case, the packet sent by user **1** would be partially affected by the packet sent by user **2** but from the right side.

Whenever either Markov or CC code set is implemented within a scenario as described so far and any two codewords are considered from the whole set, it will be possible for a packet to experiment only partial collisions but only from one side at a time. This is justified by code construction. The same however, does not happen in case of Memoryless code, where, again by construction, two consecutive packets can be transmitted at the same frequency. In this sense, for instance, up to three packets would be compromised in a similar situation. Another interesting aspect is that if two or more sequences are running under the same clock drift direction, for instance, users **1** and ***K***, they will keep the initial desired level of cross-correlation property throughout the whole radio control session. Obviously, this will have a significative impact on the system performance, as we stated earlier.

Based on what has been previously analyzed, we performed the collision analysis in two stages at every network activity period during the simulation session. At the first level,

we analyze for collisions within the current network activity period (e.g. for example in the time gap ranging from 40 to 60 ms in Figure 3.8). It will comprise possible full and partial collisions within one network activity period. In this case, a partial collision is verified if hopping patterns are unaligned, as would be the case for packets transmitted by users **1** and **2** within the time period from 40 to 60 ms, for instance. This first stage, which deals only with one single network activity period, is satisfactorily solved by simply evaluating the same condition as in equation (3.7), which is applied systematically throughout all the active users within the same period of time under analysis (i.e., a given network activity period). This situation is indicated as **(1)** in Figure 3.9.

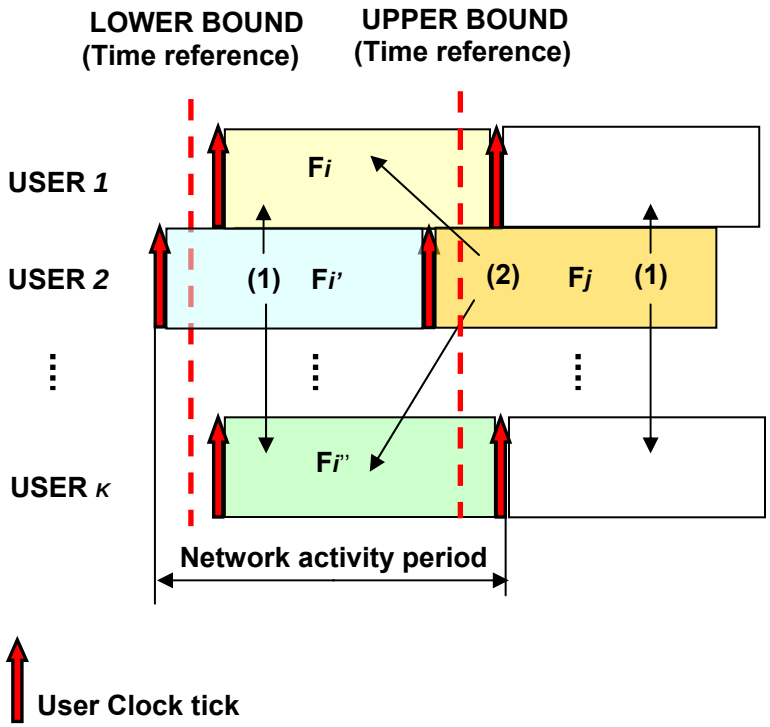


Figure 3.9 Kernel of collision analysis for SFHSS-MA with real clock case

What has been considered here as a second stage of the collision analysis will always involve the current network activity period plus the consecutive one. The analysis is now focused on the interaction between any given two consecutive network activity periods. In this case, we are looking for possible RF emissions coming from different users that overlap in time and frequency with each other; something that will also give rise to partial packet collisions. As part of this analysis, users within the current network activity

period whose clock's next tick is greater than the upper bound associated with the same network activity period are specially identified and tracked during the simulation session. Figure 3.7 will help to illustrate this kind of analysis. For example, users **1** and ***K*** transmitting at frequencies F_i and $F_{i''}$ respectively are classified under this kind of *special users*. Related with this, we store the user's number, the actual channel in use, and the value of time the clock ticks for those users (if any) in separated variables. In the case of the last parameter, it is stored by default for all the active users within any given network activity period. These values are kept on track and updated at every network activity period.

At this second level of collision analysis, we attempt to solve for typical situations where the event of a partial collision from the left (considering both packets transmitted at frequencies F_i and $F_{i''}$, respectively) with respect to the packet that is transmitted at frequency F_j for instance (indicated as **(2)**) could normally happen. For this aim, we systematically perform a comparison of the value of time the clock ticks (i.e., user message period value) for each user with the rest of the active users in the same network period. For example, based on the same figure, the status corresponding to the packet that is transmitted by user **2** within the rightmost network activity period at frequency F_j could be affected by packets that were transmitted by user**s** **1** and ***K*** at the previous activity period. In fact, these packets overlap in time, even when they belong to two different activity periods. Note that these packets started to be sent from the previous network activity period using frequencies F_i and $F_{i''}$, respectively. An overlap in time and frequency will give rise to packet collision as mentioned before.

The very next value of the clock tick at user **2** with respect to the one associated with the use of the RF channel at frequency $F_{i'}$ results to be smaller in value if compared with the same clock timing value at user **1**. As a consequence of this, packet at frequency F_j could be potentially damaged by the packet that was sent from the previous activity period at frequency F_i, if the RF channels match. Something similar involving users **2**

and ***K*** could also happen. In this case, the opposite happens with respect the previous example, since the time value of user ***K*** results to be bigger than the corresponding to user **2**. Again, a partial collision from the left with respect to the packet that is to be sent over the RF channel at frequency F_j by user **2** could potentially happen. These different situations are verified whether the following condition individually holds:

$T_c^i > T_c^j$	(3.12)

Or vice-versa, where the term T_c refers to the message rate at any given generic user *i, j*.

At this level of collision analysis, we need to evaluate for **two conditions** in the following order within a given network period. First, we verified whether or not the above-mentioned inequality holds. Depending on the nature of the previous inequality, the *a posteriori* collision analysis will be performed in two different manners. For the value of sub-index *i*; with $i \in \{1,2,...,numbUsers\}$, we evaluate the whole set of active users for sub-index *j*, where $j \in \{1,2,...,numbUsers-1\}$.

After condition given by equation (3.12) is verified, then condition given by equation (3.7) is checked for RF frequency matching as usual**.** If both conditions hold, then all the packets involved are declared corrupted. As stated before those packets belong to different network activity periods, since what really matters is the interaction between any two consecutive activity periods. Depending on the result of this double evaluation, the sub-set of declared erroneous packets may vary.

By systematically performing the above two levels of analysis, we solve for both collision patterns. It is important to notice that in case of CC codes, for instance, given that users **1** and ***K*** clocks drift into the same direction, packet sent at frequency F_j would be partially affected by only one packet from the left (either at F_i or $F_{i''}$) from the whole set of codes. This is due to the fact that all the sequences which are hopping under a

same-behaved clock will hold the original orthogonal property during the whole radio session. This fact may or may not happen in case of hopping sets based on Markov and Memoryless sequences. As a consequence, more complex collision schemes where three packets are involved could give rise in these cases.

In previous development, such as SFHSS with ideal clocks, our interference model was implemented in a simplistic way. A two-state random generator was associated with each of the channels belonging to the hopping band. The rate at which a given channel appeared corrupted depended entirely on the value of the *PER*; that, of course, in turn depends on the channel *SNR* by means of the *BER*.

Within this specific scenario, we have slightly modified our model of interference by taking into account the Adjacent Channel Interference (ACI) phenomenon. A general view of the interference scenario (in its most complete version) that we try to conceive here is shown in Figure 3.10 [35].

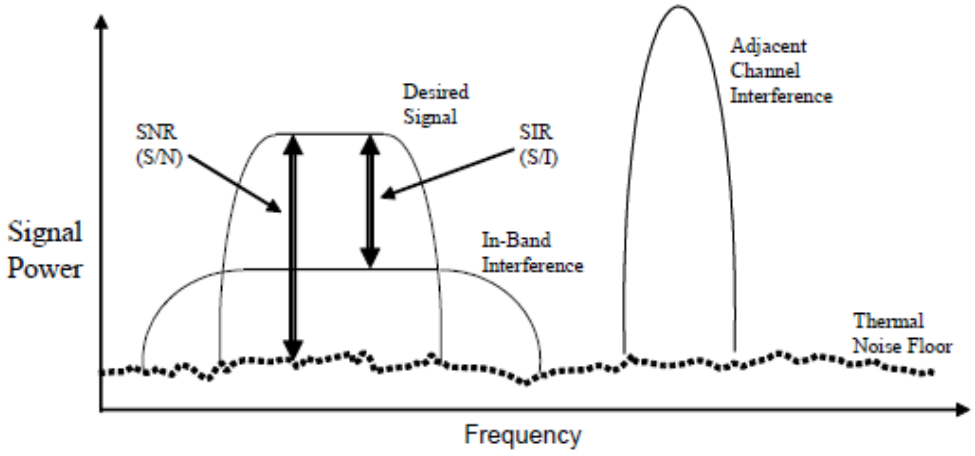


Figure 3.10 Interference scenario for a typical wireless ISM application [34]

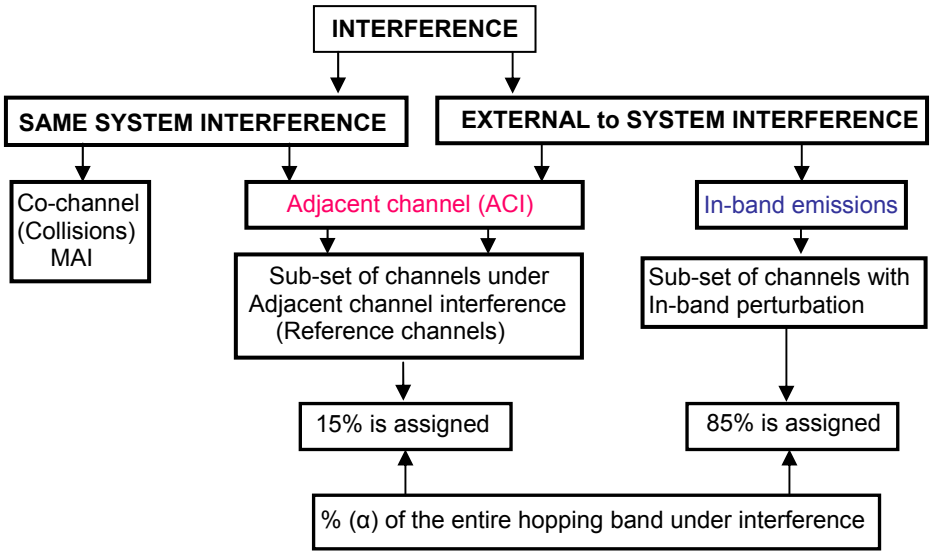


Figure 3.11 General interference model

Based on what is shown in Figures 3.10 and 3.11, we implemented our general model for interference. A percent of the entire hopping band that will be affected by interference is first defined. This percent will correspond to the parameter α mentioned in Section 3.1. It is then split based on the concept of interference source by assigning a given weight to both, the ACI, that could come from same system or sources that are external to the network, and to the in-band interference phenomenon which in this case is only associated to sources that are external to the network. A weight of 85% is assigned to the in-band interference and 15% to the occurrence of ACI at any time. The latter percent will define the so-called *reference* channels (i.e., victim channels that belong to our network) when dealing specifically with this type of interference. The percent of occurrence that is assigned to each of the interference source is something that is flexible, depending on the environment where communications take place. We in general believe that the percent of affection due to in-band emissions is more likely to be higher than the ACI.

Related to the ACI analysis, we have considered a frequency scheme ordering such that indexes that are contiguous in a typical hopping table (as shown in Figure 2.1) will also imply real RF frequency values that are adjacent in this case. This simple assumption will simplify the frequency proximity analysis within the narrowband interference analysis.

Based on this, when looking for corrupted packets due to this kind of interference, we only check for adjacency of order unity (below or above) with respect to the current channel under analysis.

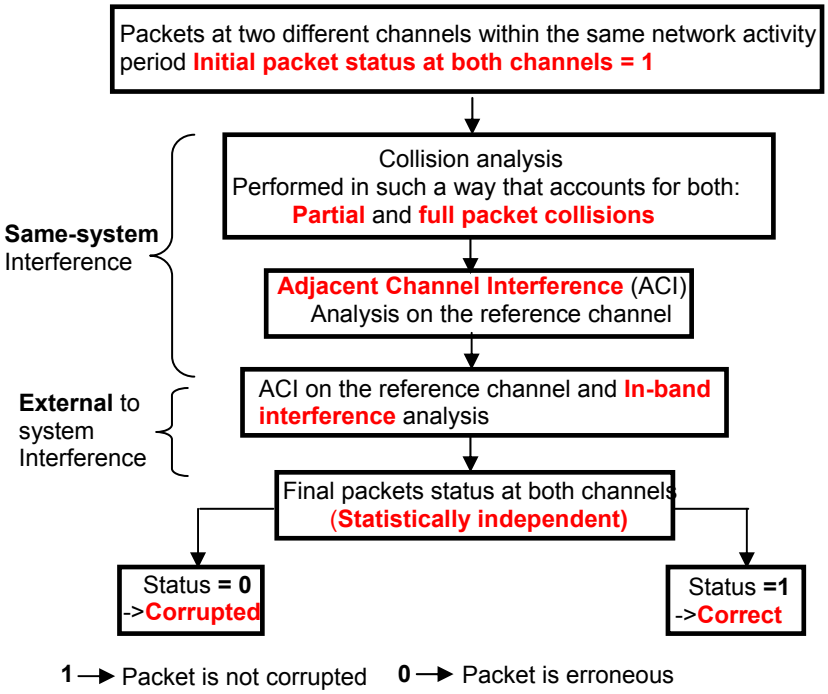


Figure 3.12 Algorithm for the collision and interference analysis in the slow SFHSS-MA scenario with real clocks

As can be seen in Figure 3.12, the analysis when considering ACI is combined with the collision detection routine performed for this specific scenario. Being a particular network activity period under such an analysis, we first check the presence of collision as explained earlier in this section. Again, the condition given by equation (3.7) is used exclusively within a single given network activity period when checking for collisions. However, the combination of equations (3.8) and (3.12) is employed when the interaction of any two consecutive network activity periods due to clock drift is of main interest. If one of the previous conditions does not hold within its respective scenario (i.e., full or partial collision), then we check for channel proximity as shown in Figure 3.12. In this case, if the targeted channel appears as being affected by ACI (recall that we consider this channel from the ACI point of view as the reference channel) we declare the packet that was sent at this channel is corrupted if the adjacency order is one. In case in which the

order of the adjacency is not one, but the targeted channel still appears affected by ACI type, we declare the packet corrupted, assuming for this case that the interferer is a source that is external to our network.

In this particular scenario, it is assumed that the transmitter and the receiver clocks (timers in each of the master-slave pairs in the network) are kept in synchronism as much as possible as it is achieved by the modified transmitted reference algorithm [17]. For this aim, the transmitter side is continuously sending timing data using special beacons or by embedding it within a normal low-level protocol frame (indicated as **Time Sync** in Figure 3.13). The transmitter sends a time stamp that corresponds to a reading of its own clock when the synch information is sent to the receiver in each or almost every packet (it could be flexible). The receiver, in turn, compares the received time data with its internal timer and will proceed to adjust any difference between the readings. The scheme we followed here for the receiver synchronization can be found explicitly in [17].

In case the receiver did not get the correct time data (because the intended packet was corrupted, indicated as event **(B)** in Figure 3.13), the very next tick will be clocked with respect to the current value it has in memory, which corresponds to the timing data contained in the last correctly received packet. In the example, such a reference is contained in the packet received at time instant equal to 200 ms. As a consequence, the actual clock value at the receiver may differ from the transmitter. Timing values obtained under proper reception circumstances are normally stored in the receiver subsystem as part of the modified transmitted reference algorithm [17]. Of course, as soon as a packet is correctly received, both clocks will highly match. What has been explained so far, applies straightforwardly when it happens that N consecutive corrupted packets are received, in which case the receiver will eventually go under acquisition stage as stated before.

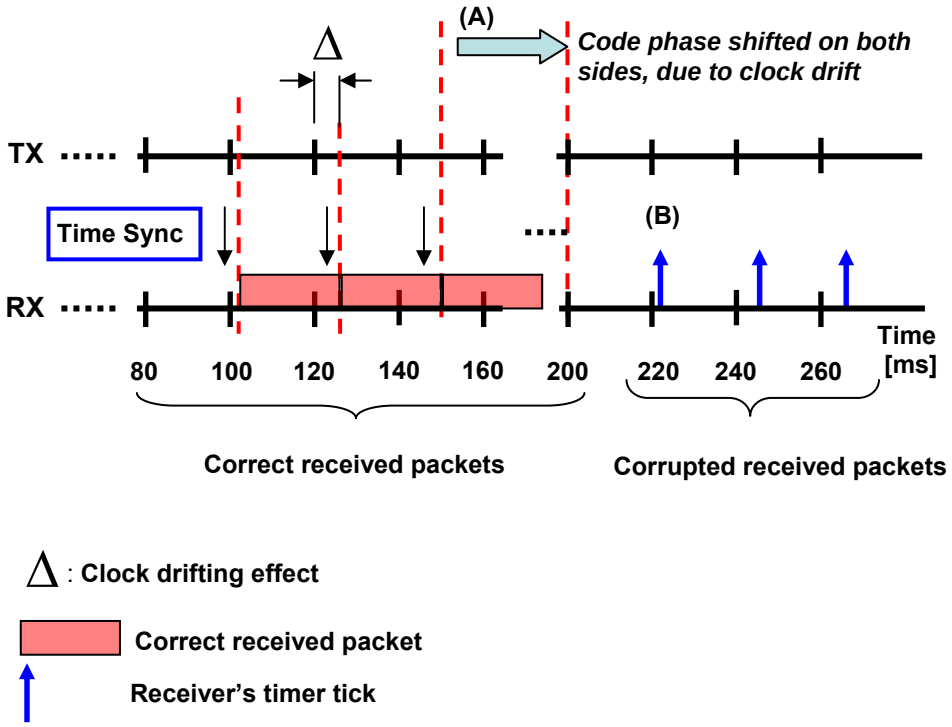


Figure 3.13 Transmitter-Receiver data flow timing diagram with real clock

We recall here that the receiver under acquisition process will keep hopping, but with the difference that its dwell time (only under this circumstance) will be much longer with respect to its intended transmitter. This has been adopted as the strategy to follow for the receiver being under acquisition stage (i.e., while searching for time synchronism) throughout this work [10], [11]. In this situation, the receiver timer is still clocking using the last good time reference it has [17]. As soon as the system re-acquires (i.e., acquisition phase ends), both clocks will run quite synchronously to each other and both (transmitter and corresponding receiver) start hopping at the same rate.

Initially when both the transmitter and the receiver units are powered on, acquisition phase is commonly achieved. As we assumed independent clocks in each of the units within a given master-slave association, the actual clock period at the receiver may differ from its intended transmitter. It is something that makes our model more realistic. In the related literature is commonly considered that the receiver system when powered on wakes up at a determined frequency value or hopping sequence phase that may differ from the corresponding value at the transmitter [15]. Related to this, we assumed, as

explained before, that by procedure, the receiver will be tuned at the midpoint of the allowed hopping band. Again, once synchronism is successfully attained, the receiver timer will get the in-lock condition (i.e., timers' value matching condition) with the transmitter, and the tracking process takes over as normally expected. From this moment forward and during the time the lock-in condition lasts, whenever a correct packet is received, the clock at the receiver will tick at the same rate as the transmitter end; also, a timing reference is updated at the receiver's memory. Under this same situation and whenever a corrupted packet is received (as is indicated in the example diagram within the range from 220 to 260 ms), the receiver will keep in tracking stage but clocking at the last good timing reference.

3.4 Asynchronous Frequency-Hop Spread Spectrum (AFHSS-MA) Scenario with Variable Packet Duty Cycle

In this case, what mainly characterizes the situation is that there is no synchronization between the users at the hopping level [27]. This means that the hopping patterns are not aligned in time, as shown in Figure 3.12. Users will basically join the network and access the medium without following any coordination or discipline at different delays with respect to each other.

A basic principle lies behind this new scenario: If two users hop to the same channel, they may or not collide. However, as found in our previous development (i.e., SFHSS-MA), if two users hop to the same RF channel at the same time, they will always collide with one another.

The diagram in Figure 3.14 shows a simplified scene of a multi-user scenario where just three users have joined the network in the time interval that has elapsed from 0 (taken as a time reference) to 60 ms.

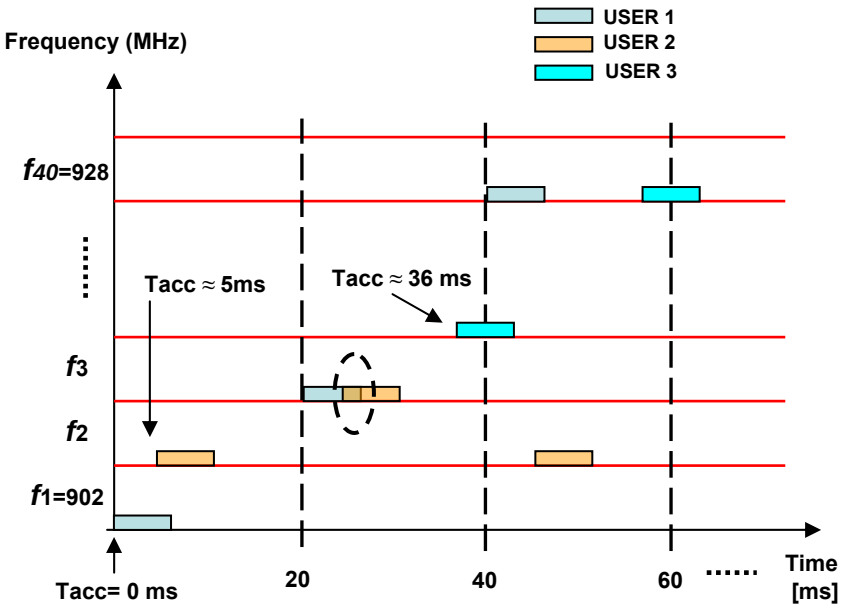


Figure 3.14 Asynchronous FHSS-MA scenario. Packet duty cycle: 30%

As an essential temporal parameter, it has been indicated the *access time or the time a given user joined the network* (T_{acc})**.** For instance, user **1** joins the network at time $T_{acc} \approx 5ms$ while user **3** does at time $T_{acc} \approx 36ms$.

We considered a fixed data packet duty cycle of 20 ms in our previous modeled scenarios developed in Sections 3.2 and 3.3. This meant that a given user had a time slot equivalent to 20 ms to transmit the packet. In this way, the medium (i.e., the channel at which the user is supposed to access) will be *strictly busy* or occupied for that period of time. We have already evaluated our system performance under this condition, something that specifically concerned the SFHSS-MA case.

System performance could be improved by featuring an adjustable data packet duty cycle capability of the transceiver. It has typically been pointed as one of the general solutions for coexistence in a multiple-access wireless networks context.

It is well-known the advantages and drawbacks that imply varying the packet duty cycle. For instance, when dealing with either intra-system (mutual interference) or cross-network interference, reducing the duty cycle will help by allowing for more coexistence.

This is commonly achieved by increasing the data rate or through data compression techniques [38]. Operating a system at a duty cycle less than 100% will certainly reduce the time that frequency sub-band or channel is being occupied. Note that in the event of two users hopping to the same frequency, the amount of time they will interfere with each other would just be the dwell time. Also, from the consumption point of view, transmitting at higher data rates will make lower the current in transmit mode [43]. However, implementing longer duty cycles will allow for more physical distance range between the transmitter and receiver. Usually, a lower transmission data rate will allow for better sensitivity at the receiver for the same physical range. As a consequence, the system throughput will certainly be increased [43].

It is quite common to find in the technical specifications of the ISM transceivers some flexibility regarding the possibility for varying the data rate. For instance, the CC1101 from Texas Instruments Inc. offers a range from 1.2 to 500 kBaud [3].

In this development, we have combined two main aspects: the no synchronization at the hopping level, in addition to the possibility of some ISM transceivers for varying the packet duty cycle by means of changing the data transmission rate.

3.4.1 Timing Parameters

In the asynchronous case, care has been taken when considering timing aspects for system performance analysis. We have already mentioned one of them, which is *the time a user joins or access the network for the first time*. For this purpose, we generate a vector of random values of access time for the total users that will be in the network. We bounded these values to certain limits in time with respect to the duration of the simulation session. The total simulation time is closely related to the total number of packets to be sent, usually set for more than 1000. The access time of the very first user is always assumed to be zero.

The timing diagram depicted in Figure 3.15 is useful in the understanding of the rest of the time parameters, their meanings and relationship. Again, and as in previous

developments, we have considered the vertical dashed lines parallel to the frequency axis as *temporal references* to which the *relative delay* of a given user could be referred. This is something used in the simulation process to allow users to join the network as their access times have already been generated. The number of active users in the network is gradually increased as their access times fall in a given network activity period as the simulation time is running. For instance, in the period of time between 40 and 60 ms, users **2** and **3** joined the network at different frequencies (which depends on their hopping patterns). This is indicated by the corresponding (T_{acc}) parameter.

Based on the same diagram, we defined the *relative delay***,** indicated by (Δt_x), as the relative amount of time between a user T_{acc} and the lower bound that corresponds to the network activity period in which this user has joined the network. It is given by:

$$\Delta t_x^i = T_{acc}^i - \text{Lower Bound}_j \tag{3.13}$$

Where $i \in \{1,2,...,numberActiveUsers\}$ and $j \in \{1,2,...,numberNetworkActivityPeriods\}$

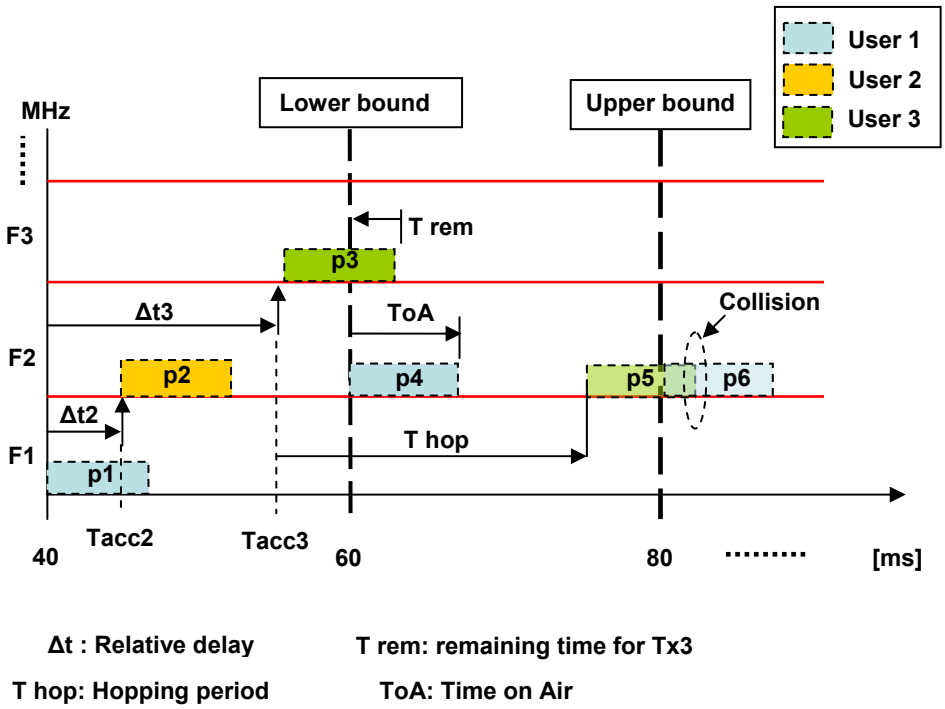


Figure 3.15 Timing diagram supporting the asynchronous model

For instance, if user **2** joined the network by accessing the medium for the first time at the time instant $T_{acc} \approx 44\,ms$, then its relative delay will be of around 4 ms from then on, in virtue of the periodicity of the sequential hops that are performed by the transmitter. We associate a unique value of relative delay to each user throughout the simulation session once a user joins the network (the reason for this is that we assumed that only one user can join the network one at a time). From the same diagram, for example, the relative delay associated with user **3** is greater than the previous one, since this user randomly joined the network at time $T_{acc} \approx 55\,ms$.

The *ToA* or dwell time**,** which was already introduced in previous developments, is closely related to the data packet duty cycle. It is defined in the initialization block of our program and is kept fixed during the whole session. There is the possibility for a user to send the packet in less time than 20 ms and this will give the opportunity for another user(s) to potentially use a free time slot if they access the medium at the same channel. However, the physical channel condition (i.e., noise effects) will remain the same as it

was conceived in the synchronous case. This is one of the variables of our problem to play with when evaluating system performance in this new condition.

The other important parameter is the remaining time (T_{rem}) a given transmitter will be active beyond the upper bound in a given network period. It is given by:

$$T_{rem}^{i} = (T_{acc}^{i} + ToA_{i}) - \text{Upper Bound}_{j} \tag{3.14}$$

Where $i \in \{1,2,...,numberActiveUsers\}$ and $j \in \{1,2,...,numberNetworkActivityPeriods\}$

From Figure 3.15, it can be seen that this situation will hold always for user **3**. This obviously has certain implications as similar kinds of users will always finish transmitting in the next network period. This special case is carefully tracked in our processing engine, as is shown next.

3.4.2 Collision Analysis

The way interactions between users behave is constantly being monitored as the aforementioned time sub-divisions (network activity periods) succeed in time. Users start joining the network and hopping periodically following respective hopping sequences. In our model, we build a new hopping sequence as a new user joins the network. This new sequence is stored together with those that belong to the already existing active users.

As we are working with a less percent of data packet duty cycle with respect to what was done in the synchronous case, the core of the collision analysis for this new situation is somehow different. Based on the temporal parameters presented in Section 3.4.1, an RF emission overlapping between two users is verified if besides condition given by equation (3.7), the following condition holds:

$$\Delta t_{x}^{j} < \Delta t_{x}^{i} + ToA \tag{3.15}$$

In equation (3.15) we have that $i, j \in \{1,2,...,numbActiveUsers\}$ and $(i < j)$.

In the collision analysis, the RF channel matching condition is evaluated first. In case that it is satisfied, condition given by equation (3.15) is evaluated next. If it is also satisfied, packets corresponding to users *i* and *j* are declared corrupted automatically. For instance, based on the example diagram, there is no collision registered within the period from 40 to 60 ms, even when condition given by equation (3.15) holds for users **1** and **2**. The reason is because these users are transmitting at different RF channels.

Users that satisfy the following condition are considered under a special kind of users:

$$T_{rem}^{i} > 0 \tag{3.16}$$

Where $i \in \{1,2,...,numberActiveUsers\}$. This is the case of user **3** referred to the same diagram. As we mentioned earlier, we take special care with this case in order to evaluate all the possibilities that lead to packet clash in this asynchronous scenario. When the period of time that elapses from 40 to 60 ms is analyzed, some new events happen. In this case, two new users have joined the network, and one of them will remain in transmission within the next activity period (60 to 80 ms). For this purpose, we implemented a set of two variables storing the following values per user (reserved only for this kind of user):

1. Current user RF channel
2. Amount of time the corresponding transmitter will be on air (which is the *remaining time* mentioned before)

Within the period from 80 to 100 ms, users **1** and **3** meet at the same RF channel. In this case, the second kind of collision will happen, in virtue of the conditions given by equations (3.7) and (3.17), respectively:

$$\Delta t_{x}^{i} < T_{rem}^{j} \tag{3.17}$$

Where $i, j \in \{1,2,..., numbActiveUsers\}$ and $(i < j)$.

As a consequence, if packet **p5** was good up to this point (began being transmitted between 70 and 80 ms), it is now declared corrupted as it will be packet **p6**, as shown in the diagram in Figure 3.13. The *a-posteriori* update that follows the collision checking is in essence the same as it is performed in the synchronous case (presented as five points at the end of Section 3.1). The noise random generator is invoked at every network activity period, similar as it is done in the synchronous case. The system metric performance (*SLOP*) is computed, as usual, at the end of the simulation session as per user basis.

Chapter 4

Experiments setup and Simulation Results

In this chapter, we validate our model framework that was explained in the previous chapter for both the SFHSS and AFHSS-MA scenarios. It is important to bear in mind that the receiver and channel models framework have been basically the same for both scenarios. We used the same following practical expression as suggested in [11] to compute our main system performance metric (*SLOP*) at any given user in the above study cases:

$$SLOP_i = \frac{\text{Total number of lags}_i}{\text{Total number of correctly received packets}_i} \tag{4.1}$$

As we constantly register the inter-arrival time for correct received packets during every single simulation session, it is convenient from a practical point of view to estimate the above probability based on the concept of relative frequency. The number of lags per user in equation (4.1) is computed based on the inter-arrival time of correct packets. We basically count "ticks" that make up the inter-arrival time if the system goes under acquisition stage; the time it takes to re-synchronize is also taken into account for this purpose. If the inter-arrival time is such that it exceeds the value assumed for *HRT* (assumed in this work as around 100 ms), a lag will be registered.

Several simulation sessions were carried out in order to obtain experimental results according to each of the scenarios. A number of 30 repetitions were usually carried out for all the experiments, which meant the same number of simulated single radio control sessions. A number of 60×10^3 packets to be sent (generated) per simulated session was normally adopted as well; this will define the simulation time for one single RC session. In the experimental set presented in Section 4.1, we evaluate the SFHSS-MA network with ideal clock oscillator scenario. Section 4.2 then addresses AFHSS-MA with ideal

clocks. Finally, Section 4.3 deals with the case of SFHSS-MA where a targeted real-time RC application is operated under a real clock oscillator. For all the cases, the main goal is to evaluate system performance under different operating conditions given by imposing realistic system parameter values and external to system anomalies. In relation to this, it is important to get a satisfactory level of understanding in regards the grade of relationship between system performance (targeted real-time RC FHSS based application) and intrinsic system parameters not only at the system level, but also considering the impact of a networking environment (i.e., effects of increasing the network load).

4.1 SFHSS-MA Scenario with Ideal Clock Oscillator

The first set of experiments is used to evaluate the behaviour of the proposed SFHSS network model as compared to [11]. As a result, and based on those five outputs mentioned in Section 3.1, the system performance was estimated by means of *SLOP* using equation (4.1). The value at every point of each curve should be considered as an average over different combinations of FH code sets. The hopping sets were built following the generalized random process principle and are regenerated at the beginning of each of the simulated sessions.

The experiments described here were performed with the aim of obtaining an individual system performance (referred to any given user/application in the network) by means of the *SLOP,* when perfect synchronization at the hopping pattern level is considered at any given time instant. For this purpose, system design parameters such as N and the probability of successful detection (P_d) were considered. Recall that N is associated, as explained before, with the modified transmitter reference synchronization algorithm. In both experiments, the network load and the number of distinct hopping carriers were fixed to 15 active users and 40 hopping frequencies, respectively.

The results of the first experiment show the dependency of *SLOP* with the *PER* as the latter is varied, shown in Figure 4.1. Three values of parameter N (3, 5 and 7) were taken into consideration:

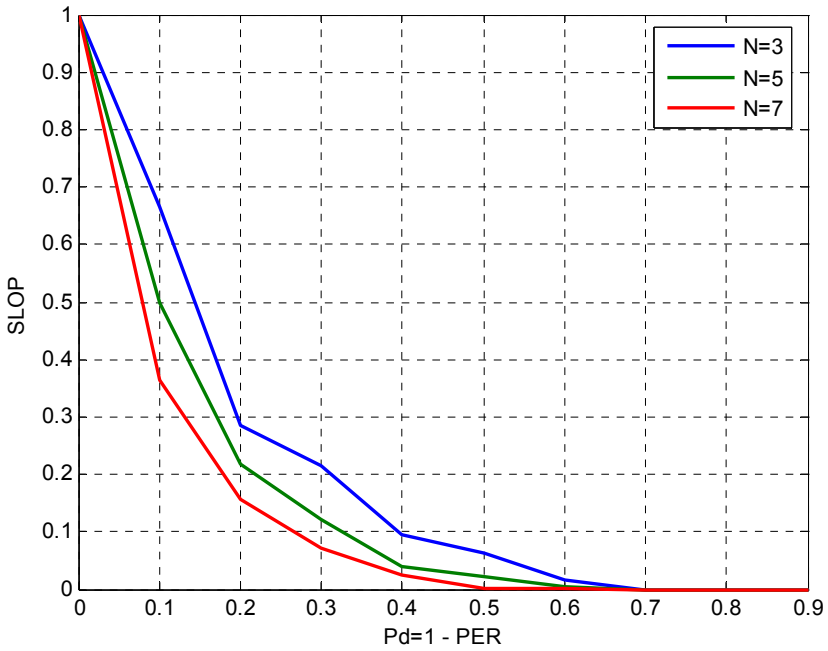


Figure 4.1 *SLOP* vs *Pd* for SFHSS-MA with ideal clock scenario (15 users and 40 RF channels)

Within this section, system *PER* is mainly associated with the harmful effect of RF emissions by sources that are exogenous to the system; we refer specifically to the well-known in-band interference phenomenon caused by other systems that operate in the ISM band. In general, the presence of that kind of interference and intra-network collisions (i.e., MAI) are the most claimed factors when considering performance degradation at this point.

By simple inspection of graphs in Figure 4.1, it is possible to observe a similar behaviour for an individual system (from one user/application perspective) if compared with results in [11]. For instance, as the value of parameter *N* is increased, the single system performance improves being the *SLOP* below 0.1 (or virtually zero if parameter *N* is set to seven) when half of the hopping band is affected by interference other than MAI. This fact will also be verified in the results corresponding to the next experiment.

Theoretical curves shown in Figures 3.4 and 3.5 could be used to conclude that for the dependency of the *SLOP* on *PER*, a better system performance is possible to be attained for the same level of interference occupancy (α), the same network load, and the same

value of N if a greater number of distinct hopping carriers (q) are considered in the network. As we mentioned earlier, memory requirements increase linearly with the number of frequencies considered for hopping [9]. This could lead to a trade-off situation in respect to the system performance, as the number of available hopping frequencies and system *SLOP* are inversely proportional to one another.

The second experiment relates *SLOP* directly to the value of parameter N for a fixed value of interference occupancy (50, 70, and 90%). Graphical results are shown in Figure 4.2.

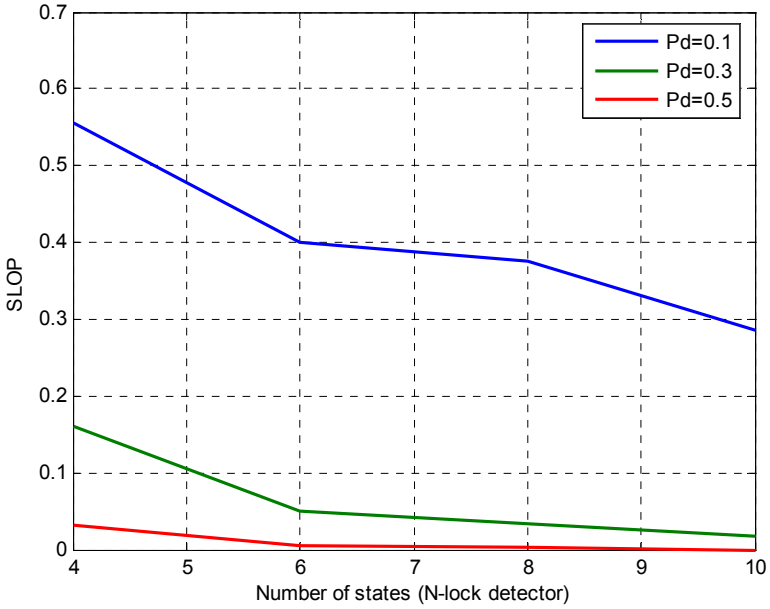


Figure 4.2 *SLOP* vs *N* for SFHSS-MA with ideal clock scenario (15 users and 40 RF channels)

In this case, the main idea is to evaluate a feasible alternative from the system design point of view, with respect to the common solution of increasing the number of hopping carriers to alleviate system performance from the harmful effect due to partial-band interference and MAI. In fact, as shown in the results, as the value of parameter N was increased the *SLOP* decreased; this was expected based on the previous experiment. Both sets of results are in complete agreement with one another. It is interesting to note that for a network load of 15 users, 40 hopping frequencies, $N = 5$, and with half of the entire

hopping band under interference, it is possible to attain a satisfactory level of *SLOP* of less than 0.1.

Despite the similarities observed in our results with those obtained in [11], certain differences were expected to be found since in our model, no specific channel or a set of them is pre-defined to be blocked 100% of the time. In our case, the *blocked* condition used in [11] for a given channel was given in our case by the chance for a collision to happen between hopping sequences that are generated. It is known that this is subject to a certain probability, as explained in Section 3.2.1.

In relation to this set of experiments, it was possible to verify the similarities of our results with those delivered in [11] concerning the behaviour of the performance for a single user (real-time RC application) that operates in a MA scenario when ideal clocks and generalized random stationary hopping patterns are modeled. The effectiveness of increasing the value of the parameter *N* in order to alleviate the degradation effects due to interference as a whole is evident.

4.2 AFHSS-MA Scenario with Ideal Clock Oscillator

In this section, we present the results of our second set of experiments corresponding to the AFHSS-MA network. The impact of considering a variable packet duty cycle on system performance is evaluated mostly with the *SLOP* metric. This fact is considered of critical importance under this new scenario due to its asynchronicity nature. As mentioned in Section 3.4, it has typically been addressed as one of the general solutions for coexistence in a multiple co-located users wireless networks context.

For the first experiment conducted here, we defined a given percent of packet duty cycle and then kept it fixed during the simulation session. The packet duty cycle was set to 30%, which represents six milliseconds (with respect to 20 ms assumed as the nominal dwell time thus far). The network load was fixed to up to 15 active users and the number of distinct allowed hopping channels to 40. The value of *N* was taken as three in order to evaluate for the worst case scenario in regards to the receiver synchronization algorithm.

Based on the above-mentioned settings, system performance was evaluated as a function of the data packet duty cycle.

The value at every point of each curve should be considered as an average over different combinations of FH code sets. The hopping sets were built following the generalized random process principle and are regenerated at the beginning of each session.

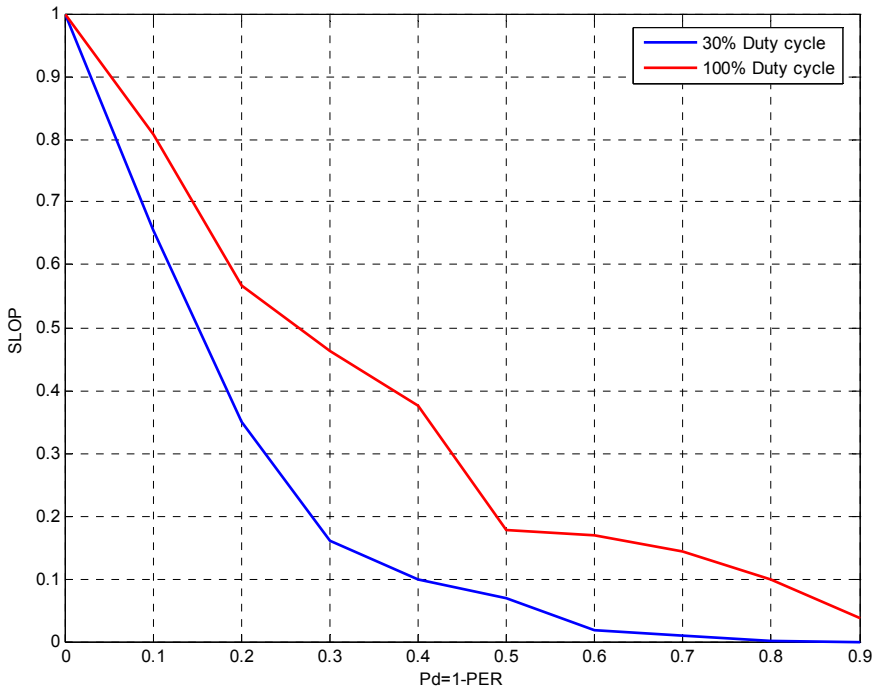


Figure 4.3 *SLOP* vs *Pd* for AFHSS-MA with ideal clocks (15 users and 40 RF channels)

As shown in Figure 4.3, one can determine at first glance how the duty cycle would play an important role in getting better system response (as mentioned before within this section). The values of *SLOP* that were obtained for a single user/application being operated at 30% of the packet duty cycle resulted to be lower than those corresponding to a full cycle for the whole range of *PER*. To be more precise, a *SLOP* value of less than 0.2 was obtained for a single user/application operating at 30% of the data packet duty cycle, being the interference occupancy as high as 70%. This result was expected in that the real chances for collisions are decreased as the channel occupancy is reduced. To this effect, as the receiver has to go under acquisition or resynchronization process less often

because corrupted packets are less frequent to happen, the chances for a lag to occur are lower as well (recall that for a lag to happen, the inter-arrival time of the packet has to be greater than the *HRT*). This fact makes the *SLOP* in equation (4.1) lower. Even if the receiver system undergoes a re-synchronization, the chances to acquire before 100 ms have elapsed are higher than before because the probability for a packet to be received erroneously is lower.

The results corresponding to the second experiment, depicted in Figure 4.4, feature the dependency of *SLOP* as the packet duty cycle is varied in a wide range. For this purpose, a 50% for the interference occupancy (α) was chosen, the network load was fixed to 15 co-located users, the number of hopping carriers was fixed to a typical value of 40 RF channels, and the *N* parameter was accordingly set to three.

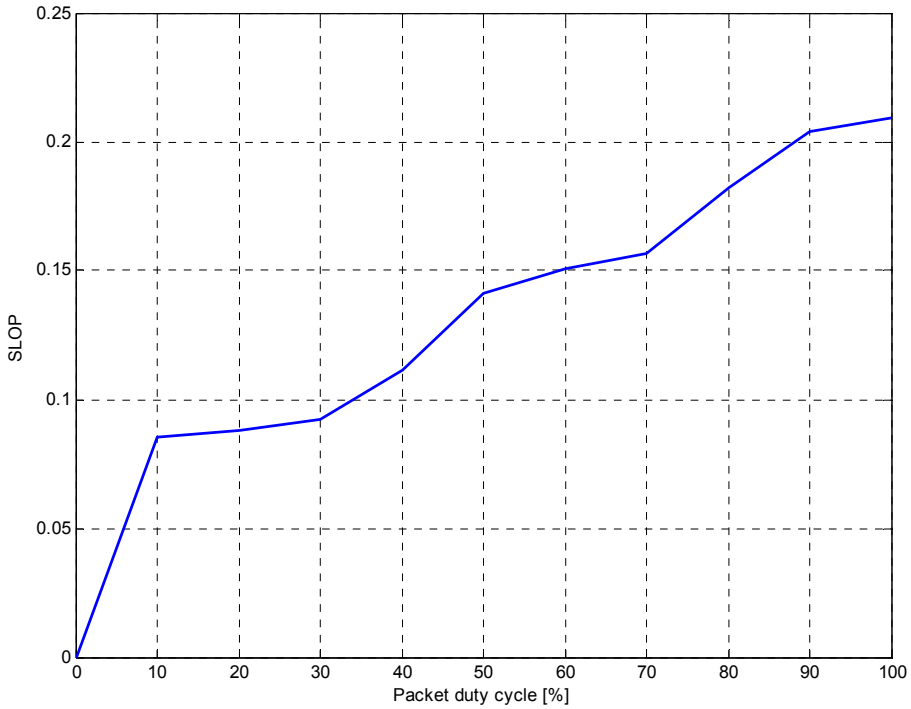


Figure 4.4 *SLOP* vs packet duty cycle (15 users, 40 RF channels, and α=50%)

Based on the previous figure, it is possible to observe how the response of a single user/application will worsen as the data packet duty cycle at which it is operated is increased. As stated before, the longer any given transmitter occupies the channel (i.e.,

dwell time), the chances for it to be interfered or hit by another user that belongs to the same network are more likely.

For values of packet duty cycle lower than 30%, for example 10%, the *SLOP* was very low. Based on that value of packet duty cycle and a typical packet size of 160 bits, a data transmission rate of 80 Kb/s is achieved. The feasibility of this example is given by how ISM transceivers as the CC1101 of Texas instruments can perfectly handle data rates of 250 Kb/s.

A comparison of both graphs shows how the *SLOP* values were very similar to each other for 30% of the data packet duty cycle and 50% of interference occupancy $(P_d = 0.5)$. This fact demonstrates consistency between those experiments that were conducted adapting our model framework to this scenario.

4.3 SFHSS-MA Scenario with Real Clock Oscillator

The fact of having RC SFHSS systems driven by real clocks is considered a critical issue in this research when considering system performance evaluation. It is our aim to solely estimate typical hopping codes performance first and then evaluate system performance through its direct application to our slow SFHSS-MA scenario, where real clocks are considered for all the active users in the network. The part that is specifically dedicated to evaluate FH codes performance is found in the Appendix A. The second part of this set of experiments, shown in this section, is closely related to the results shown in the appendix. It addresses the direct application of each of the family of FH codes to our main model framework, which comprises the channel (communication medium) and transmitter - receiver models under the real clock assumption.

4.3.1 SFHSS-MA System Performance with Real Clock

This section covers a vital part of this work in that we consider all the features that will make our scenario as close as possible to resembling real life. In the initial experiments that have been analyzed within Section 4.1, we obtained an idea of the system performance by means of the *SLOP* where we assumed a perfect synchronism at the

hopping level. In this more pragmatic case, certain asynchronicity at the hopping level is introduced by considering imperfect clocks. In modeling for a typical clock oscillator, we have followed all the principles mentioned in Section 3.3.

The involved clocks were initially set at a given output frequency that is randomly generated from the two possible values given by equation (3.9). The clock drift rate was also calculated based on the same concept given by the previous equation and is also given by equation (3.11). The accumulative error due to clock deviation is accounted by parameter (Δ) given by equation (3.10).

In this section, we present results based on our main system performance metrics: the well known *SLOP*, and the Probability of Losing a Packet denoted here as (PoLP) in general. *PoLP* at user i^{th} has been defined for this purpose as:

$$\text{PoLP}_i = \frac{\text{Number of corrupted packets}_i + \text{Number of missed packets}_i}{\text{Total number of packets sent}_i} \tag{4.2}$$

As seen from equation (4.2), not only have the total of corrupted packets been taken into account, but also those that have been rejected (as the acquisition mechanism assumed through this work operates) when the phase of the locally generated long FH sequence does not match with the received version.

There are several variables to play with: the number of active users or network load; the number of RF hopping channels; the number of states *N* in the modified transmitted reference synchronization algorithm; the fact of considering the channel impairments (specifically, external to the system interference); clock accuracy; and, of course, the type of FH pattern that is employed in the network.

This experimental part is divided into two sections which target different metrics in order to evaluate an individual user/application performance. The first section is dedicated to estimate an individual system performance through the *PoLP*. In pursuing this, we did

not consider any kind of noise interference perturbing an individual system other than MAI. The second set of experiments addresses the *SLOP* as a metric to be used, since the effects of interference sources that are external to the system are also considered together with MAI. In this particular case, an individual system performance is evaluated when parameters intrinsic to this kind of system are varied, such as the number of hopping channels, the kind of FH code that is implemented, the network load, and the clock initial accuracy or tolerance.

Regarding the *PoLP*, a first experiment will evaluate system behaviour when only competition for the medium between users is of interest, subject to the implementation of different kinds of FH code sets. In this aspect, we varied the network load from 5 to 60 co-located users. The clock accuracy was set to 100 ppm. The value of the number of corrupted packets the receiver system will handle before re-synchronizing in time under the modified transmitted reference synchronization algorithm (N) was set to three. Every single simulated RC session, as usual, was ran for 60×10^3 packets (that roughly represents 20 minutes based on approximately 20 ms for the dwell time). Specifically, the same test was performed for the following three different hopping patterns: Markov, Memoryless, and those based on the theory of Cubic Congruences. The value at every point of each curve should, in general, be considered as an average over different combinations of FH code sets and clock shift patterns. In the case of the CC code set, as it is deterministic, no average is specifically considered over distinct code sets.

As we used 40 RF hopping channels for this experiment, it is important to point out that due to the fact that, in case of the CC code set, it is theoretically impossible to generate more than 40 users for the same number of hopping channels; we needed to consider the same number of RF channels for all the code sets beyond the 40 RF channel test as was imposed by the CC case. For instance, when a number of 50 users was tested, CC codes theory establishes that exactly 52 hopping frequencies will be generated and, of course, the network load could be kept as 50 users. Then a fair combination of 50 users-52 RF channels is used for the three types of code sets. Results are presented in Figure 4.5.

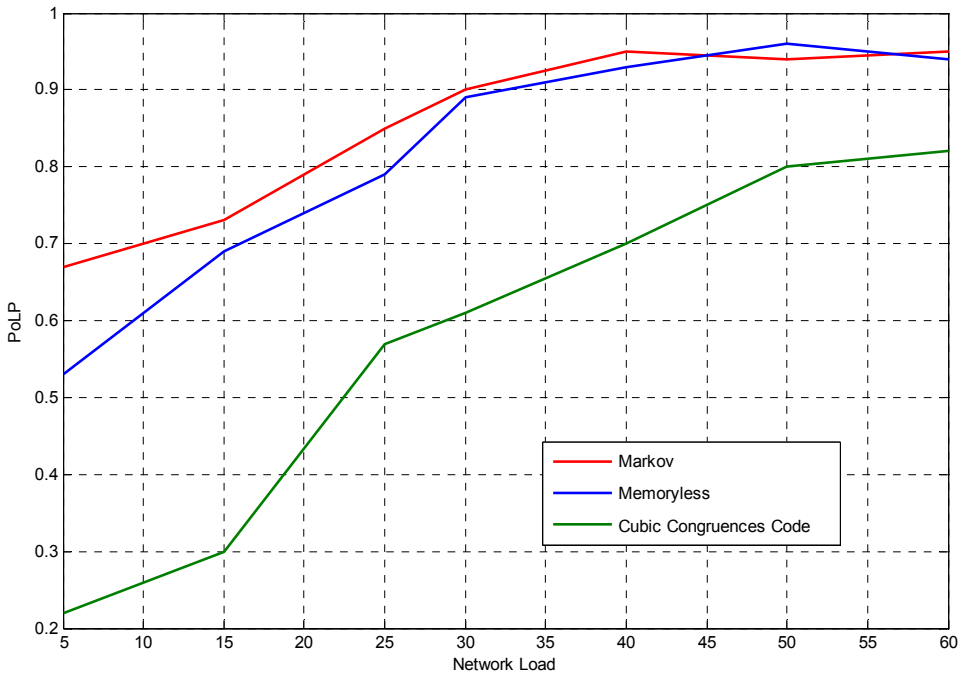


Figure 4.5 System *PoLP* vs the network load for three different FH code sets (*N*=3 and no external to system interference was considered)

Based on the results shown for an individual system within a SFHSS-MA network that is operated with a real clock oscillator, the difference in performance is first evident when using general random stationary process or a deterministic type. This is supported *a-priori* by the auxiliary experiments performed directly on these FH code sets; where only the CC set of codes satisfactorily met the Lempel-Greenberger optimality criterion (please refer to Appendix A). Based on these results, we suggest the use of CC set of codes in a range of network load from five to 15 co-located users, for which the system *PoLP* would be less or equal than 0.3.

It can also be seen how increasing the network load contributes to the deterioration of the system performance. This could be definitely improved for a given network load by increasing the number of FH channels to more than 40 RF channels, or the value of the parameter *N*, for example, to a value of seven. We believe that by implementing a combination of these two facts, a more balanced solution can be reached.

Another experiment related to the system *PoLP* was carried in which the aim was an evaluation of such a parameter as the radio session time elapses. In this experiment, two values for the clock tolerance were tested: 50 and 100 ppm. The network load was fixed to 15 active users and the number of distinct frequency hopping channels was set to 40. Only Markov and CC FH codes set were taken into consideration for this experiment since, for the previously mentioned value of the number of distinct hopping frequencies, the behaviour exhibited on the average by a Markov based FH code set suitably resembled the Memoryless case.

For this purpose, we divided the total amount of packet to be sent (the same amount as the previous experiment) that would correspond to the total simulation time in equally spaced time intervals or packet batches. The system loss was evaluated at the end of each of these time periods. Of course, such an operation has an accumulative effect as the simulation time passes because we are carrying the loss experimented in a given previous time interval on to the current one and so on until the end of the simulation session.

The main idea here was that, according to the loss the system is experiencing, a general re-synchronization process could be performed through the network in order to avoid increasing losses by bringing the set of hopping codes back to a position in time where the best value of the cross-correlation property is held. In case of CC code set, as shown in Figure A.1, the cross-correlation value of any given pair of hopping sequences is initially zero. In this regard, this state would be the best to attain from the whole range of relative code displacements. In general, this situation would be more difficult to determine in case of the so-called general random general processes considered here (i.e., Markov and Memoryless).

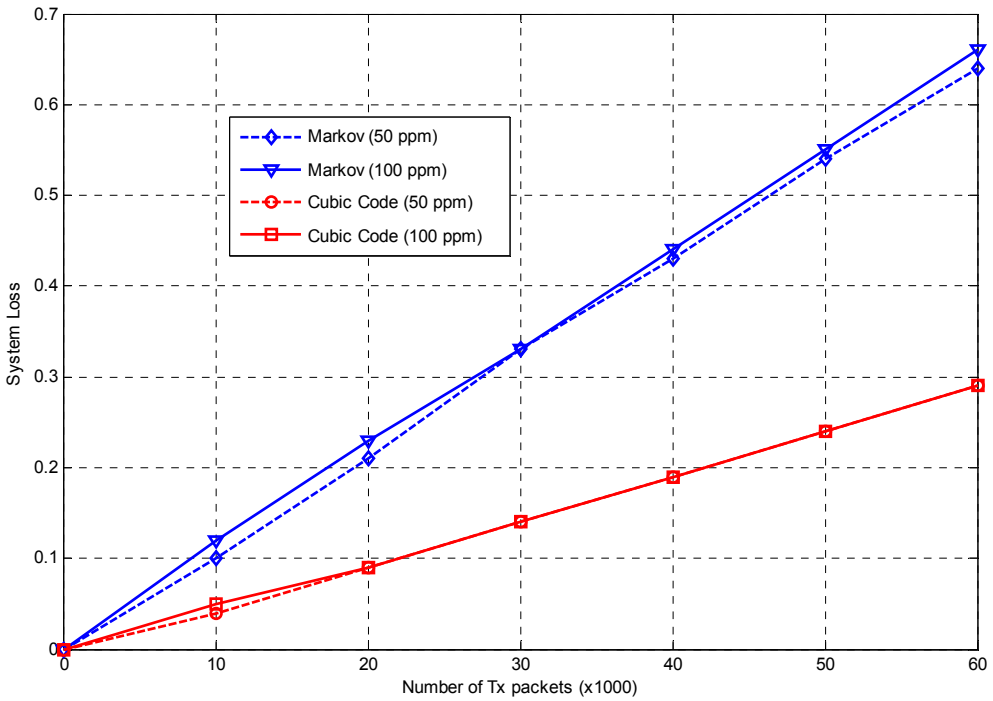


Figure 4.6 System *PoLP* vs elapsed time for two kinds of FH code sets and clock accuracy (40 RF channels, 15 users, N=3 and only MAI was considered)

Based on Figure 4.6, it is possible to clearly distinguish, again based on previous experiments, the difference in performance regarding the two kinds of FH code patterns targeted in this case. For instance, a value of *PoLP* of 0.3 is reached for the CC based code set when the elapsed time is approximately 20 minutes (equivalent to 60×10^3 packets). This fact could be taken as a practical criterion for re-synchronizing all the hopping patterns throughout the network.

It should also be noted that whatever the hopping pattern, when a less accurate clock is employed, system *PoLP* is greater for the same value of elapsed time. In addition to the fact that the system loss is accumulative, this could be related to the rate at which a given set of hopping codes is losing its good cross-correlation property as the network starts operating. As seen in the set of graphs for the Hamming cross-correlation function in case of the CC code set (see Figure A.1) that for a relative lag equal to zero between codewords, the number of coincidences throughout the code set is zero. To this effect, as each code from the network starts running under a randomly generated clock shift pattern, this property will be destroyed faster or slower as the clock shift rate dictates.

However, this fact is more complicated to conclude in case of Memoryless and Markov code sets; in these cases, no uniform upper bound is found for the Hamming cross-correlation function, being that the maximum value through the set is twice as the case of CC codes. These codes are completely random; there could be a case where at the initial state a given pair of codes exhibits a cross-correlation value that is different from zero, and as there is a shift in time domain, the number of collisions could be as high as three times this value, or even lower (see Figure A.8).

It can be noticed that for a network load of 15 users, very similar values for the system *PoLP* were obtained in both graphs associated with with the two experiments referred to above for the same total number of transmitted data packets, bearing in mind that the same number of RF hopping channels $(q = 40)$ was tested. This fact shows consistency in both the model framework and the results model shown at this point.

For the experiments concerning the evaluation of system performance by means of the *SLOP*, the value at every significative point of each curve on each the following experiment results should be considered as an average over different combinations of FH code sets and clock shift patterns. In the case of the CC code set, as it is deterministic, no specific average is considered over distinct code sets.

In contrast to the case of system *PoLP*, the interference generation block was fully activated this time in order to consider an overall *BER* according to [25], where it is proposed that two situations need to be taken into account. First, in absence of interference due to specifically: In-band and adjacent channel interference, there will still be errors due to MAI (i.e., co-channel interference coming from same system users). Secondly, there will be scenarios in which the level of collisions is negligible or zero, but errors due to interference coming from sources other than co-channel type are possible. Recall that *SLOP* depends on the overall system *PER* as it was discussed in Section 2.3.

As it follows, three experiments will evaluate not only the implementation of different FH code sets in a network, but also the impact on system performance when varying the

network load, the size of the frequency library, and the clock initial accuracy. The parameter N was, as previously, set to three in order to evaluate for the worst case scenario. The specific settings for the experiments will be subsequently described in detail.

In the first experiment, we fixed the clock initial accuracy to 100 ppm. The network load was considered for 15 active users while keeping the size of the frequency library as 40 hopping channels. The results corresponding to this experiment are shown in Figure 4.7 for the two kinds of FH code sets that have been documented within this work (i.e., general random stationary and deterministic).

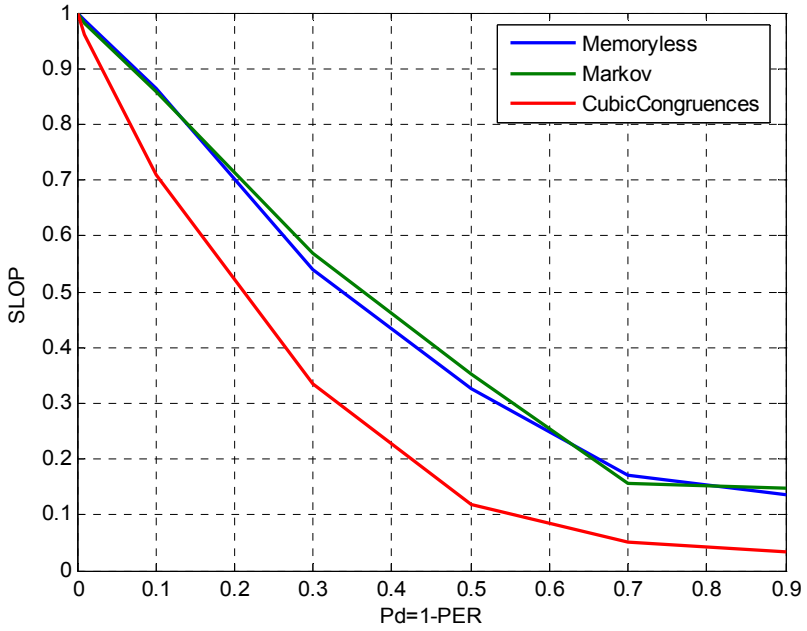


Figure 4.7 *SLOP* vs Pd for Markov, Memoryless and CC FH code sets (40 RF channels, 15 users and interference is considered)

It can be seen that the deterministic based FH code set (i.e., CC) outperformed what we could consider the Markov-Memoryless set for the whole range of interference occupancy. For instance, the CC code set can be used delivering acceptable values of *SLOP* (less than 35 %) in a critical range for interference occupancy (α) from 40 to 70 %. The Markov-Memoryless set exhibited approximately the same behaviour but for an interference occupancy range from 30 to 50 %. It should also be noted how Markov and

Memoryless based FH code sets performed fairly close to one another as the size of the frequency library was set to 40 RF channels. This means that typical Markov and Memoryless FH patterns will cause the system to behave similarly for high values of distinct hopping carriers (considering as a criterion $q \geq 40$). This was expected, as shown in the results in Appendix A.

In the second experiment developed within this scenario, we wanted to show the dependency of *SLOP* on the number of active users and the effect of the size of the frequency library on system performance. In this case, we evaluate the system performance for two values of these parameters: 15 and 30 active users and for 40 and 80 RF channels, respectively, as shown in Figure 4.8 in a composited graph.

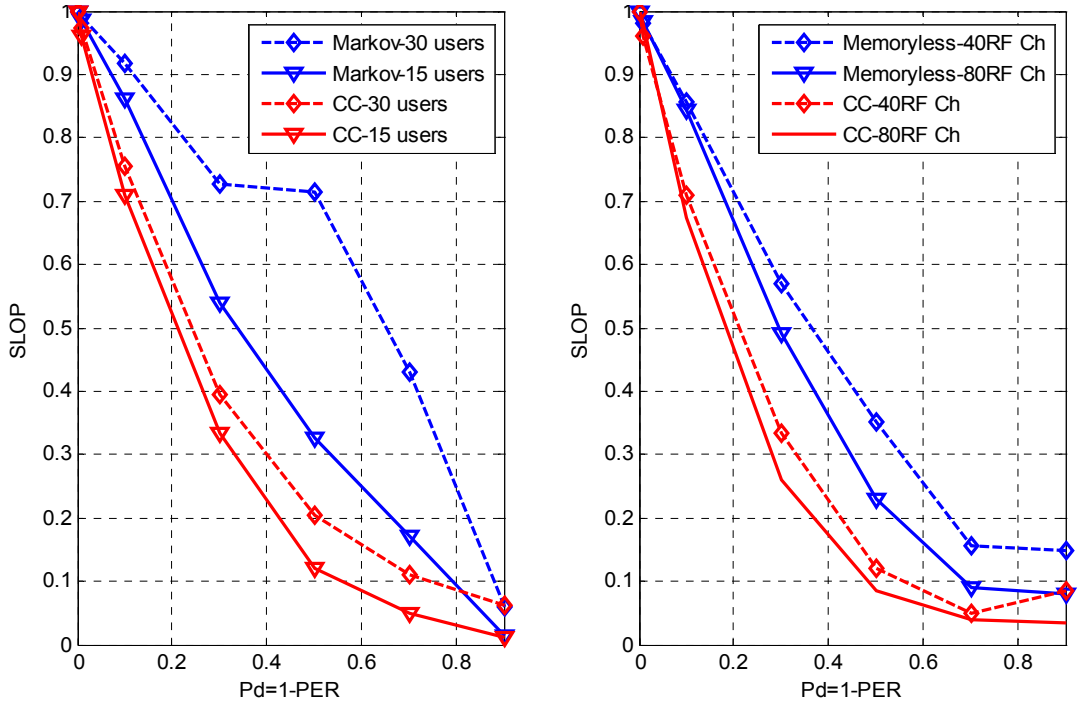


Figure 4.8 *SLOP* vs Pd. Dependency as the network load is varied (left). Dependency as the number of channels hopping is varied (right)

From results shown in the previous figure, it is perfectly clear that as a general trend, the performance of a single user/application will be worse as the network load increases with whatever kind of hopping pattern employed. On the other hand, as the number of hopping channels or the size of frequency library is increased, the system performance will improve. This conclusion was expected, based on the theoretical curves shown in Figures 3.4 and 3.5, respectively.

As stated before, the negative effects of the interference seen as a whole on system performance can definitely be alleviated by increasing the number of distinct RF hopping channels. This result was more evident in case of FH code sets based on general random processes. A *SLOP* reduction of approximately 10% was attained in case of the Memoryless FH code set for an interference occupancy of 50%, when the size of the frequency library was increased from 40 to 80 RF channels (see the rightmost graph in Figure 4.8).

At another level of comparison, specifically between kind of FH patterns, we have that the CC code set caused the *SLOP* to be less than 20% for a level of interference occupancy of 60% in the average, while by using memoryless based FH codes, it was possible to attain a level of *SLOP* less than 30% but with half of the hopping band under interference.

The last experiment deals with the dependency of the system performance on the accuracy of the clock oscillator. The evaluation was performed for two values of this parameter, specifically 50 and 100 ppm. The network load and the size of the frequency library were fixed to 30 active users and 40 channels, respectively. Results can be seen in Figure 4.9.

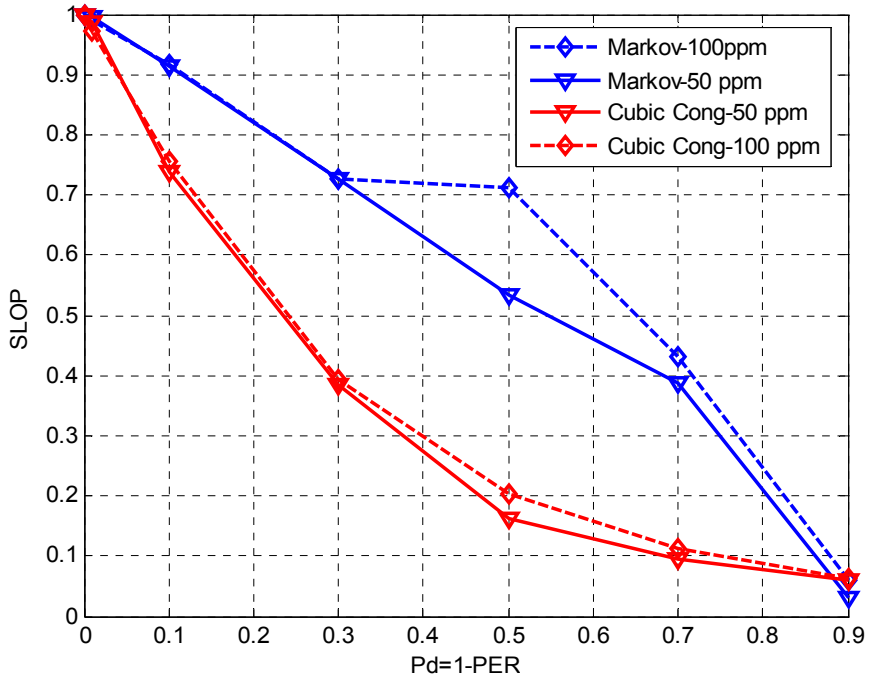


Figure 4.9 *SLOP* vs Pd. Dependency as the clock initial accuracy is varied (40 RF channels, 30 users, and interference is considered)

In relation to the performance evaluation of a single user/application within a SFHSS-MA network operated under real clock oscillator condition, this experiment studies the influence of the clock initial accuracy by means of the *SLOP*. Similar to results shown in Figure 4.6 that addresses the system *PoLP*, it was noticed that system performance was

worse as the clock initial accuracy was increased. As explained before, we believe it is possible to justify this effect based on the behaviour of the cross-correlation property of a given set of FH codes as time elapses.

As can be seen in Figure 4.9, we are simulating variability on the percentage of interference occupancy; this is a factor that obviously impacts system performance and which acts independently of the nature of the FH pattern. Based on the behaviour exhibited by the set of Hamming cross-correlation functions shown in Figure A.4 for the CC FH code set, the good cross-correlation property that this family of codes holds at lag equal to zero (as shown in Figure A.1) will be compromised as the set of codes are randomly shifted through its rows. For this specific case, in order to give rise to interference due to collisions, any two hopping sequences or codewords from the set need to be shifted in opposite directions. Operating these sequences at 50 or 100 ppm will just accelerate the possibility for more collision to happen per unit of time. Using a 50 ppm clock (based on equation (3.9) for $F_o = 50Hz \equiv 20ms$), 20×10^3 ticks of the clock will be needed for one complete time slot to be shifted; in case of 100 ppm is used, only half of this is needed. This will increase the probability of more packets to be hit per unit of time, and that will in turn increase the probability for any given system to experience a lag. Of course, as has been shown throughout this work, it is possible to compensate this effect by considering a larger value for the parameter *N*. Recall that this experiment was already conducted for $N = 3$.

In regards to the case of a FH code set based on general random processes, it is something more difficult to conclude as they are completely aleatory. This fact could also be inferred from the behaviour of the Hamming cross-correlation functions for Markov and Memoryless based FH code sets (see Figure A.7) compared with the corresponding cross-correlation functions mentioned previously.

In conclusion, the Synchronous and Asynchronous FH-MA were the main testbench where the targeted real-time RC based application on which this work is focused was analyzed. Experiments were conducted with the objective of studying the relationship

between key system engineering parameters and the performance of such an application in a new condition as the multiple access networks; this could be considered as the main global objective of this research. Based on this, we learned how the following system aspects impact the response of a real-time single RC FHSS based application: the maximum number of erroneous packets that the modified transmitted reference algorithm tolerates before re-synchronization (N); the use of real clock oscillator; the cross-correlation property of the hopping set; the variable data packet duty cycle; and the interference phenomenon in all its extension. Finally, the nature of the interaction between those parameters as seen in the experiment results definitely helped in allowing us to propose viable solutions when anomalies either external or intrinsic to the application are present.

Chapter 5
Conclusions

In this chapter, we will first return to the main contributions of this thesis. The limitations of the proposed models will then be outlined. Finally, the work proposed in this thesis can be extended in several directions. As a result, some suggestions based on our experimental results will be made not only for actual system usage, but also, for any future research that may be developed on the topic.

5.1 Overview of the Contributions

In this thesis, we developed a state-of-the-art with respect to the results presented in [11] since a characterization of the response of a single real-time RC SFHSS-based application within a *multiple user scenario* was carried on. In doing this, most commonly referred FH code sets were considered and modeled systematically to our prototype of RC network. Key aspects concerning the interaction between the transmitting and receiving entities (i.e., Primary master-slave association) for the targeted RC application were evaluated under the most recent technological mainstream. This was possible based on an exhaustive review made on actual SoC ISM transceivers data specification and application notes.

Specifically within the multi-user environment, Synchronous and Asynchronous FH-MA scenarios were individually taken into consideration. Performance evaluation through two main metrics (*SLOP* and system throughput) was conducted for a single user/application under these two networking environments. In relation to this, system engineering parameters, such as the use of real clock oscillator and the cross-correlation property of the hopping codes, were taken into account within the synchronous case. When considering the asynchronous network, a variable data packet duty cycle was featured as a key system parameter. All of this, in addition to the completeness of our model for the interference phenomenon, allowed us to evaluate a single application response under more realistic operating conditions.

5.2 Current Limitations

The *asynchronous* network scenario comprising multiple slow frequency-hop spread spectrum real-time RC applications could be considered as the more general multiple access scheme that has been analyzed in this work. Each user has been assumed to transmit L binary symbols per hop (slow FHSS) using non-coherent orthogonal FSK modulation. Based on the previous conditions, what we believe can be a limitation of our modeling framework is the fact that, in order to be exceptionally accurate at the time of evaluating system performance, two kinds of collisions should be considered according to [42]: collisions at the hopping pattern level (something that we treated in our analysis as *full* or *partial* types), and those ones happening at the FSK modulation tones.

Full and partial types of collisions were possible to be identified because of the nature of the access scheme mentioned above (i.e., due to the asynchronicity). However, the fact that either of those events occur not necessarily means that the symbol being transmitted at a certain point in time by the user under analysis would be hit or affected by the interfering one. The pseudo-random generated carriers can be the same at both transmitters (i.e., what would normally correspond to a full or partial collision as seen in our model description), but the resulting modulation signal may not be due to the modulating symbol. There is a probability governing this kind of event known as the average symbol error probability, as defined in [42]; it is a function of the probability defined by equations (3.3) or (3.4) (depending on the kind of FH pattern under analysis) and a conditional probability of symbol error when hits occur from the rest of the users in the network.

Within the context of the situation explained above, it is worthy to mention that the grade of affection when considering interference at the FSK tones level is commonly subject to an orthogonality criterion [26], [28], [42]. If the FSK signals are considered to be at the minimum isolation in order to meet orthogonality, the asynchronicity can compromise this property. However, if the separation is large enough, partial collisions will not cause interference on both of the matched filters of the BFSK demodulator [42]. This is known as the Geraniotis' assumption. The assumption of this condition (for slow hopping

scenarios as it is considered in this work) has been shown [28] to result in an optimistic bound with respect to the real symbol error probability.

The inaccuracy introduced in this work could be alleviated if we consider that for a moderately high number of hopping carriers (q), our application is such that the number of binary symbols transmitted per hop in equation (3.4) is much greater than unity. This implies that as the probability of partial hits $(qN_b)^{-1}$ becomes negligible, so does the cross-interference between FSK tones.

As a second aspect when considering limitations in the present work, in order to accurately take into account the interference coming from other same-network users when analyzing a given targeted user, the respective signal power levels (i.e., at every FSK tone) should be differentiated as opposed to considering them the same. The relative distances between co-located users should be modeled in somehow as it will make our MA scenario even more realistic.

Finally, in regards to the transmission medium, we believe that a complete channel characterization must include the effects of the nonselective and selective types of fading; specifically the class of nonselective Rayleigh fading channel and the selective wide-sense stationary uncorrelated-scattering fading channel respectively.

5.3 System Configuration Guidelines and Future Work

We have learned the following from our experiments about the effectiveness on system performance of varying certain engineering parameters: the number of distinct hopping carriers; the maximum number of erroneous packets that the modified transmitted reference algorithm tolerates before resynchronization (N); the type of FH pattern employed; the packet duty cycle; and the accuracy of the clock oscillator.

It is of our knowledge the range of values of those key parameters mentioned above that can provide an acceptable level of satisfaction for standard or low level RC real-time applications as those featured in this work (i.e., model cars and aircrafts in the hobby

category). Based on this, for typical values of the number of hopping channels and network occupancy of 40 RF channels and 15 active users, respectively we recommend the following; if synchronicity is achieved at the FH pattern level through the network by means of the use of extremely accurate clock oscillators with initial accuracy of the order of ± 1.5 ppm, the CC FH family of codes should definitely be employed. This fact will help to maintain the level of collisions very low for a time period corresponding to 20 minutes of radio control session as simulated in the experiments. Under these circuntansces, the code shift will be in this case of 30 ns per clock tick considering that this happens around of the desired value of 20 ms. Recall that CC code sets exhibit zero number of collisions at zero shift in time domain, this will cause that interference due to MA be virtually null. In the ideal case of perfect synchronism as considered in [11], there will not be interference due to MA.

However, such a value of clock accuracy belongs to a high performance clock which may be quite expensive and therefore, not justifiable for the kind of real-time application we are dealing with. If cost becomes a constraint, typical ISM SoC's clock accuracy of around ± 60 ppm will still cause an adequate average packet loss close to 5 % per10^4 data packets that are to be sent. Constrains impossed in practice will lead to an optimal configuration for the set of system parameters to be used, this means that other parameters such as the number of hopping channel and the number of states (N) in the transmitted reference synchronization algorithm could be varied even when a good class of FH pattern is to be employed.

In the more general case (i.e., AFHSS-MA scenario with ideal clocks) the use or not of deterministic FH patterns will not exceptionally improve the system performance since the best cross-correlation property status would not hold due to the random delays between users accesing the medium. For similar system configuration (number of hopping channels and network occupancy) a reasonable data packet duty cycle value of 30 %, employing typical FH patterns based on general random processes, will cause an acceptable system performance as the *SLOP* would be below 0.2 for an interference occupancy of 70 % of the entire hopping band. Even considering a heavy interference

scenario of half of the hopping band under interference, if an excesive reduction of the packet duty cycle becomes a constrain, there will not be major problems since for a range up to 60% of the duty cycle it is still possible to attain an acceptable level of system performance ($SLOP \leq 0.2$). We recall as previously, that this is something that always should be seen in conjuction with the possibility of varying other system parameters such as the number of hopping channels and the number of states (N) in the modified transmitted reference algorithm.

For future work to be done on the topic, we recommend the implementation of an acquisition mechanism other than the *uniform serial search enabled by a matched filter* for a slow-FHSS system, while trying to keep the flexibility of the *modified transmitted reference synchronization algorithm* for the receiver. This is due mainly to the relative long acquisition times that we experimented in our simulations at this stage. This will definitely impact the performance of the targeted application as the probability for a lag occurrence will decrease.

However, this may not be enough for a more serious application such as military, even when a highly accurate clock is implemented. We believe that in such a case, another alternative for the re-synchronization process will be highly desirable.

References

[1] Remote-controlled aircraft seller's website: http://www.rcroll.com. Last checked in Summer 2009.

[2] Remote-controlled aircraft seller's website:http://www.bananahobby.com. Last checked in Summer 2009.

[3] "CC1101 Low-Power Sub-1GHz RF Transceiver". Data Sheet. Texas Instruments Inc., 2009.

[4] K.H.Torvmark,"Frequency Hopping Systems", Application Note AN014. Chipcon Products from Texas Instruments Inc, March 2002.

[5] "ATR 2406, Low-IF 2.4 GHz ISM Transceiver". Datasheet. Atmel products, December 2008.

[6] "nRF24E1, RF Transceiver with Embedded 8051 Compatible Microcontroller and 9 Input, 10 Bit ADC". Product Specification. Nordic Semiconductor, March 2006.

[7] "nRF24LU1 2.4GHz Single Chip Transceiver with USB Microcontroller and Flash Memory". Product Data Specification v 1.1. Nordic Semiconductor ASA, 2008.

[8] "CC2511Fx Low-Power SoC (System on Chip) with MCU, Memory, 2.4 GHz RF Transceiver, and USB Controller". Product Data Specification. Texas Instruments Inc., 2008.

[9] S. Rao, "Implementing a Bidirectional Frequency Hopping Application with TRF6903 and MSP430". Application Report. SWRA041, September 2004.

[10] "Frequency Hopping Techniques", Application Note No.51, Micrel Inc., June 2006.

[11] A. Ismail, "Performance of Real-Time Remote Control Systems in the ISM Band", Carleton University, M.A.Sc Thesis, December 2008.

[12] B. Sklar, "Digital Communications, Fundamentals and Applications", 2nd Ed. Prentice-Hall Inc., Upper Saddle River, N.J, 2001. ISBN: 0-13-084788-7.

[13] Yahya, O. Sidek, and J. Mohamad-Saleh, "Design and Develop Wireless System Using Frequency Hopping Spread Spectrum", Engineering Letters, 13:3, November 2006. Published on-line in the International Association of Engineer website.

[14] T. Vanninen, H. Saarmisaari, M.Rustia, and T. Koskela, "FH-Code Phase Synchronization in a Wireless Multi-Hop ADHOC Network", milcom, MILCOM 2006, pages1-7.

[15] D. Torrieri, "Principles of Spread Spectrum Communications systems". Springer Science + Business Media Inc., 2005. ISBN: 0-387-22782-2.

[16] V. P. Ipatov, "Spread Spectrum and CDMA: Principles and Applications", John Wiley & Sons Ltd, Chichester, England, 2005. ISBN: 0-470-09178-9 (HB).

[17] F. Dominique and J.H. Reed, "Robust Frequency Hop Synchronization Algorithm", Electronics Letters, Vol. 32, No.16, August 1996, pages 1450-1451.

[18] J.G. Proakis, "Digital Communications", 4th Ed. The McGraw-Hill companies Inc., N.York, 2000. ISBN: 0-07-232111-3.

[19] Elezabi and E. Sourour, "Performance of Hybrid Direct Sequence-Slow Frequency Hopping Spread-Spectrum Acquisition under Partial Band Interference and Fading Channels" IEEE Vehicular Technology Conference, Sept 2005, pages 1509-1513.

[20] Wilde, "Extended Tracking Range Delay Lock Loop", IEEE International Conference on Communications, 1995, Seattle, WA, USA, Vol. 2, pages 1051-1054.

[21] L. Peterson, R. E. Ziemer, D. E. Borth, "Introduction to Spread Spectrum Communications", Prentice-Hall, N.J, April 1995. ISBN-10: 0024316237. ISBN-13: 978-0024316233.

[22] "CC1000 Single Chip Very Low Power RF Transceiver". Product Data Specifications. Chipcon Product from Texas Instruments Inc., February 2007.

[23] K.T.Heien, T.A.Lunder, and K.H.Tovmark," Oversampling and data decision for the CC400/CC900", Application Note AN008. Chipcon Products from Texas Instruments Inc, February 2006.

[24] W. Golomb and G. Gong, “Signal Design for Good Correlation: For Wireless Communication, Cryptography, and Radar”, Cambridge University Press, 2005. ISBN: 9780521821049.

[25] E. A. Gerianiotis, “Error Probabilities for Slow-Frequency-Hopped Spread-Spectrum Multiple-Access Communications over Fading Channels”, IEEE Transactions on Communications, Vol. COM-30, No.5, May 1982, pages 996-1009.

[26] K. Cheun and W. E. Stark, “Probability of Error in Frequency-Hop Spread Spectrum Multiple-Access Communication Systems with Noncoherent Reception”, IEEE Transaction on Communications, Vol.39, No.9, September 1991, pages 1400-1410.

[27] K-W. Cheun, “Analysis of Asynchronous Frequency-Hop Spread-Spectrum Multiple-Access Network”, a dissertation submitted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy, Electrical Engineering: Systems in the University of Michigan, 1989.

[28] K. Choi and K. Cheun, “Performance of Asynchronous Slow Frequency-Hop Multiple Access Networks with MFSK Modulation”, IEEE Transaction on Communications, Vol.46, No.2, February, 2000, pages 298-307.

[29] T.A. Gulliver and S.H.C. Ting, ”Dual-Tone MFSK for Frequency-Hopped Spread Spectrum Multiple Access Communications”, Contemporary Engineering Sciences, Vol.1, 2008, no.4, pages 177-191.

[30] J. R. Bellegarda and E. L. Titlebaum, “Time-Frequency Hop Codes Based Upon Extended Quadratic Congruences”, IEEE Transactions on Aerospace and Electronic Systems, Vol. 24, No.6, November 1988, pages 726-742.

[31] S. V. Maric and E. L. Titlebaum, “Frequency Hop Multiple Access Codes Based Upon the Theory of Cubic Congruence”, IEEE Transactions on Aerospace and Electronic Systems, Vol.26, No.6, November 1990, pages 1037-1039.

[32] Ismail, “Performance of Packet-Based FHSS Radio Control Systems”, Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada. *This paper is pendant to be published.*

[33] R.Khalili and K.Salamatian, "Evaluation of Packet Error Rate in Wireless Networks", Proceedings of MSWIM'04, 2004.

[34] R. Gummadi, D. Wetherall, B. Greenstein, and S. Sesham, "Understanding and Mitigating the Impact of RF Interference on 802.11 Networks", ACM SIGCOMM Computer Communications Review, Vol. 37, issue 4, pages 385 - 396, 2007. ISSN: 0146-4833.

[35] "The Effects of Adjacent Channel Rejection and Adjacent Channel Interference on 802.11 WLAN Performance", Texas Instruments, SPLY005, November 2003.

[36] Kraimeche, "Effect of Interference on a Wireless Link with Frequency Hopping", Wireless Communications and Networking Conference, 2003. IEEE 2003. Vol.1, pages 132-137. ISSN: 1525-3511.

[37] Kraimeche, "Analysis of Interference in a Wireless Link with Frequency Hopping", Proceedings of the 10th IEEE International Symposium on Modeling, Analysis, and Simulations of Computer and Telecommunications Systems, 2002, pages 303-312.

[38] "CWNA™ Certified Wireless Network Administrator Official Study Guide", 3rd Ed, McGraw-Hill/Osborne, California, U.S.A, 2005. ISBN: 0-07-225538-2.

[39] "nRF24AP1 2.4GHz Single Chip 2.4 GHz with Embedded ANT protocol". Product Data Specification. Nordic Semiconductor ASA, 2006.

[40] Vectron International official website http://www.vectron.com/index.htm. Last checked on Fall 2009.

[41] "Crystal Oscillator Design Considerations", White paper, Nordic Semiconductor, November 2008.

[42] Gerianiotis, "Multiple-Access Capability of Frequency-Hopped Spread-Spectrum Revisited: An Analysis of the Effect of Unequal Power Levels", IEEE Transaction on Communications, Vol. 38, No. 7, July 1990, pages 1066-1077.

[43] "nRF24L01 Single Chip 2.4GHz Transceiver". Product Specification. Nordic Semiconductor ASA, 2007, pages 867-879.

[44] G. Ge, Y. Miao, and Z. Yao, “Optimal Frequency Hopping Sequences: Auto and Cross-correlation Properties”, IEEE Transactions on Information Theory, Vol. 55, No. 2, February 2009.

Appendix A

Frequency-Hop Sets Experiments

FH patterns play a key role on the system performance. This section is dedicated to obtain auxiliary results, based on an appropriate tool, regarding the performance of the most known FH code sets generation principles: Memoryless, Markov, and those based on the theory of Cubic Congruences (CC). In order to evaluate code performance, we built a subroutine called **Code Performance Tool** where a given matrix of codes that has been generated is considered as an input variable for this routine. These matrices are considered here as FH code sets. The rows in any of them constitute hopping patterns or sequences that are statistically independent with respect to each other. The aforementioned subroutine has as output variable the Hamming correlation functions (i.e., out-of-phase and in-phase auto and cross-correlation).

We found exceptionally convenient the use of the Hamming correlation approach [44] in order to submit a strong criterion regarding optimality of a hopping code set based on finite fields. The theoretical foundation for the use of the Hamming correlation function is explained as follows: given a set of sequences or hopping codes $\chi(v,F)$, where each of the sequences has length v over the finite field F that could be conceived as an ordered set of integers representing the m available hopping frequencies $\{f_0, f_1, ..., f_{m-1}\}$. The Hamming cross-correlation is then defined, considering any given two sequences $X = (x_0, x_1, ...x_{v-1})$ and $Y = (y_0, y_1, ..., y_{v-1})$ as follows [44]:

$$H_{XY} = \sum_{0 \le i \le v-1} h(x_i, y_{i+\tau}) \tag{A.1}$$

Where the shift parameter (τ) is taken in the range: $[0, v-1]$. The parameter enclosed by the sum above is such that:

$$h(x_i, y_{i+\tau}) = \begin{cases} 0 & if\ x_i \neq y_{i+\tau} \\ 1 & if\ x_i = y_{i+\tau} \end{cases} \tag{A.2}$$

Where all the operations among position indices are performed modulo *v*. In equation (A.1), if $X = Y$, the out-of-phase and in-phase Hamming autocorrelation is obtained for $\tau \neq 0 \bmod(v)$ and $\tau = 0 \bmod(v)$, respectively.

It is always desirable for a FHSS-MA network that the Hamming out-of–phase and cross-correlations be as low as possible. With respect to the autocorrelation, it is also very convenient that it be as impulsive as possible in order to minimize ambiguity at the time of acquisition for any given transmitter-receiver pair. In relation to these claimed properties, the Lempel-Greenberger optimality criterion [44] would contribute to give a complete characterization of a family of hopping codes within this context.

All that is needed in order to evaluate a family of FH codes is the computation of the two following parameters (*H*) for each of the distinct pairs of sequences *X* and *Y* [44]:

$$M(X,Y) = \max\{H_a(X), H_a(Y), H_c(X,Y)\} \tag{A.3}$$

Where $H_a(X) = \max_{1 \leq \tau \leq v-1}\{H_{XX}(\tau)\}$ is the maximum value of the out-of-phase Hamming autocorrelation for sequence *X*. The second term in equation (A.3) implies the same operation but in case of sequence *Y*. The rightmost term, $H_c(X,Y) = \max_{0 \leq \tau \leq v-1}\{H_{XY}(\tau)\}$, refers to the maximum of the cross-correlation Hamming function between the two sequences.

The Lempel-Greenberger optimality criterion for a family of codes $\chi(v, F)$ states that *a given set of FH codes is optimal if every distinct pair of sequences X, Y is an optimal pair* [44]. For this last condition to hold, the following inequality needs to be verified:

$$M(X^*, Y^*) \leq M(X, Y) \tag{A.4}$$

Where the right hand term in equation (A.4) is evaluated for the rest of all distinct pair of sequences (*X*, *Y*) that belong to the set $\chi(v, F)$. In such a case, the pair of sequence X^*, Y^* is said to be an optimal pair within the set.

We generated three sets of ten FH codes each. One of them was based on the theory of Cubic Congruences (deterministic in nature), while the other two were selected as based on a general random stationary process where the sequences are generated independently from each other (see Figures A.1 and A.2). The dimension (*v*) of the frequency library *F* was chosen to be ten. This implied that $F = \{1,2,3,4,5,6,7,8,9,10\}$. We omitted the value zero from the library for pure formalism.

$$y = \begin{pmatrix} 1 & 8 & 5 & 9 & 4 & 7 & 2 & 6 & 3 & 10 \\ 2 & 5 & 10 & 7 & 8 & 3 & 4 & 1 & 6 & 9 \\ 3 & 2 & 4 & 5 & 1 & 10 & 6 & 7 & 9 & 8 \\ 4 & 10 & 9 & 3 & 5 & 6 & 8 & 2 & 1 & 7 \\ 5 & 7 & 3 & 1 & 9 & 2 & 10 & 8 & 4 & 6 \\ 6 & 4 & 8 & 10 & 2 & 9 & 1 & 3 & 7 & 5 \\ 7 & 1 & 2 & 8 & 6 & 5 & 3 & 9 & 10 & 4 \\ 8 & 9 & 7 & 6 & 10 & 1 & 5 & 4 & 2 & 3 \\ 9 & 6 & 1 & 4 & 3 & 8 & 7 & 10 & 5 & 2 \\ 10 & 3 & 6 & 2 & 7 & 4 & 9 & 5 & 8 & 1 \end{pmatrix}$$

Figure A.1 Family of 10 FH codes based upon the theory of Cubic Congruences

$$y=\begin{Bmatrix}10&3&7&5&9&8&5&1&9&5\\7&8&10&8&2&5&10&10&5&9\\1&4&9&1&2&3&2&7&3&2\\1&8&5&10&5&5&9&6&3&7\\9&1&7&4&9&6&8&5&4&2\\2&7&4&6&2&7&4&9&9&6\\5&9&9&7&9&7&4&3&4&6\\8&4&9&6&4&8&6&5&7&7\\8&10&6&9&2&10&3&3&9&8\\2&1&9&2&3&7&3&5&1&10\end{Bmatrix} \qquad y=\begin{Bmatrix}10&3&7&5&9&8&5&1&9&5\\7&8&10&8&2&5&10&5&9&1\\4&9&1&2&3&2&7&3&2&1\\8&5&10&5&9&6&3&7&9&1\\7&4&9&6&8&5&4&2&7&4\\6&2&7&4&9&6&5&9&7&9\\7&4&3&4&6&8&4&9&6&4\\8&6&5&7&8&10&6&9&2&10\\3&9&8&2&1&9&2&3&7&3\\5&1&10&6&5&6&4&5&3&6\end{Bmatrix}$$

Figure A.2 Families of 10 FH codes based on Memoryless (left) and Markov (right) general random stationary processes

Lempel and Greenberger also developed a lower bound for the parameter *M* where for any pair of distinct FH sequences *X*, *Y* that belongs to the set $\chi(v,F)$ and with library size *m*, it holds that [44]:

$$M(X,Y)\geq\frac{\sum_{i=0}^{m}\left(d_i^2+e_i^2+d_ie_i\right)-2v}{3v-2} \tag{A.5}$$

Where d_i and e_i represent the number of times a specific frequency (*f*) from the library *F* appears in one period of the sequences *X* and *Y*, respectively. Based on equation (A.5), we estimated the lower bound for the parameter *M* for all possible pairs in the three set of codes. In case of the CC and Memoryless family examples we obtained 0.357 (for all the pairs) and 0.571 (the lowest registered), respectively. The latter set of codes exhibited a higher value for the aforementioned parameter compared with the CC set in all the cases. It is important to note here that each of the sequences of the CC code set is a type of non-repeating codeword, something that does not hold for hopping codes based on general random stationary processes, as it is the case of Memoryless and Markov. This property will definitely contribute to decrease the right hand term in equation (A.5) since the following two conditions are fully satisfied [44]: $d_0\leq d_1\leq\ldots\leq d_{m-1}$ with $d_{m-1}-d_0\leq 1$ and $e_0\geq e_1\geq\ldots\geq e_{m-1}$ with $e_0-e_{m-1}\leq 1$. By simple inspection of the matrix of CC codes in Figure A.1, it is possible to notice how such a dual condition is widely satisfied.

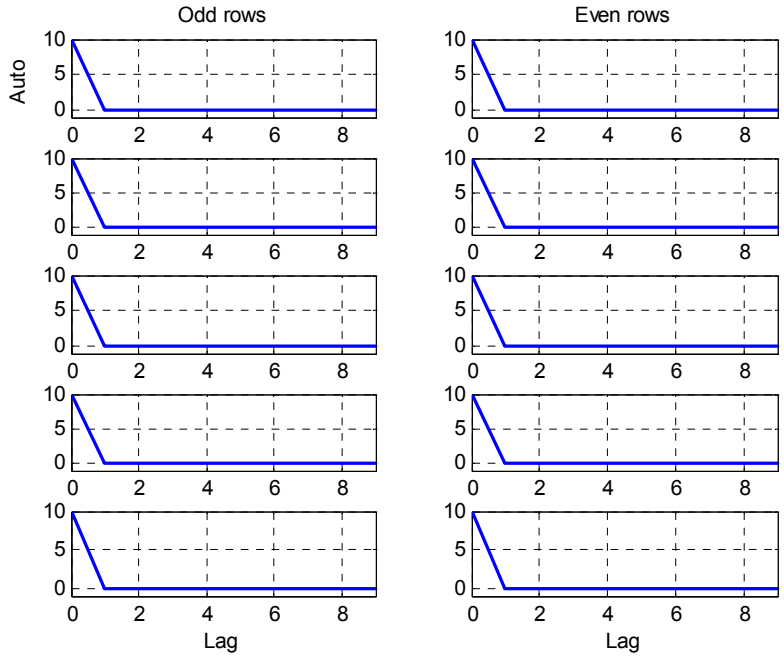


Figure A.3 Hamming autocorrelation functions for each sequence in the CC code matrix shown in Figure A.1

Figure A.3 shows the Hamming autocorrelation functions (which includes both in and out-of-phase autocorrelations values) for the CC code set presented as an example. As can be seen all the functions are quite impulsive and meet the important requirement for the out-of-phase values which resulted to be as low as zero for all the range of the delay factor (τ). On the other hand, Figure A.4 represents in this case, the cross-correlation functions for all the distinct possible combinations of row one of matrix shown in Figure A.1 with the rest of the rows.

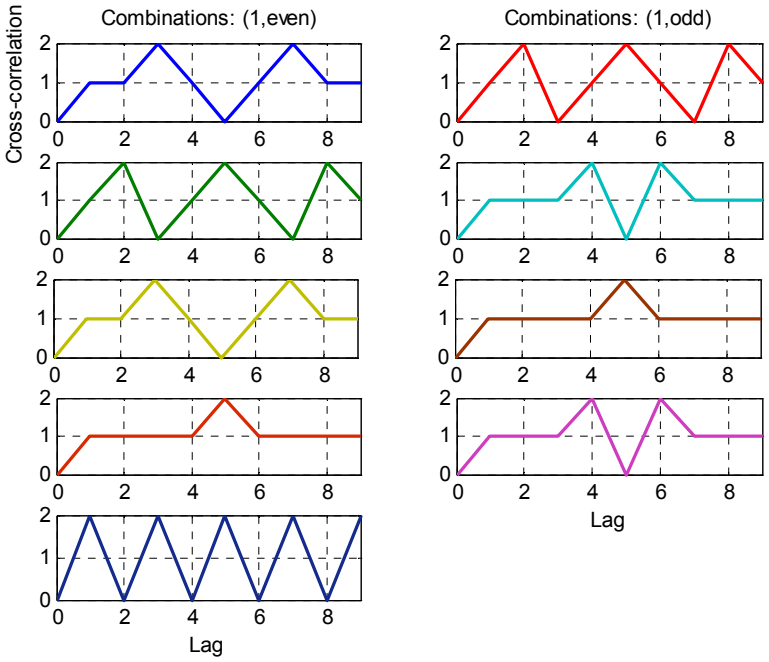


Figure A.4 Hamming cross-correlation functions for the combinations of row one with the rest from the set of CC codes shown in Figure A.1

For all the cases, a constant upper bound of as low as two was obtained, which means that when two any codewords are shifted one with respect to the other in the time domain, a maximum of two coincidences or hits are possible to occur. This behaviour was verified for all the possible distinct combinations of rows in the set.

In case of the Memoryless and Markov code sets, as shown in Figures A.5 and A.6, respectively, the shape of each of the Hamming autocorrelation function does not always meet the specifications explained earlier in this section as it was not exactly impulsive.

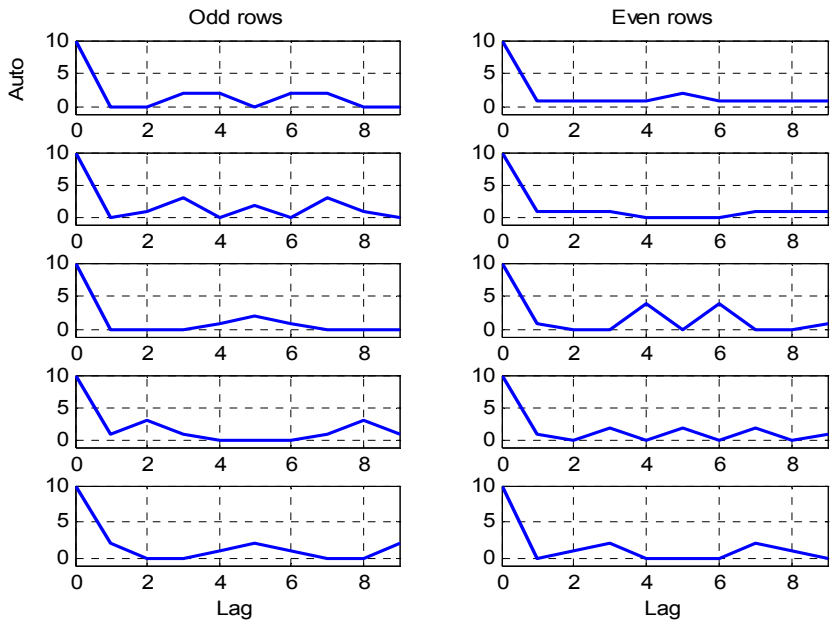


Figure A.5 Hamming autocorrelation functions for each sequence in the Memoryless code matrix shown in Figure A.2

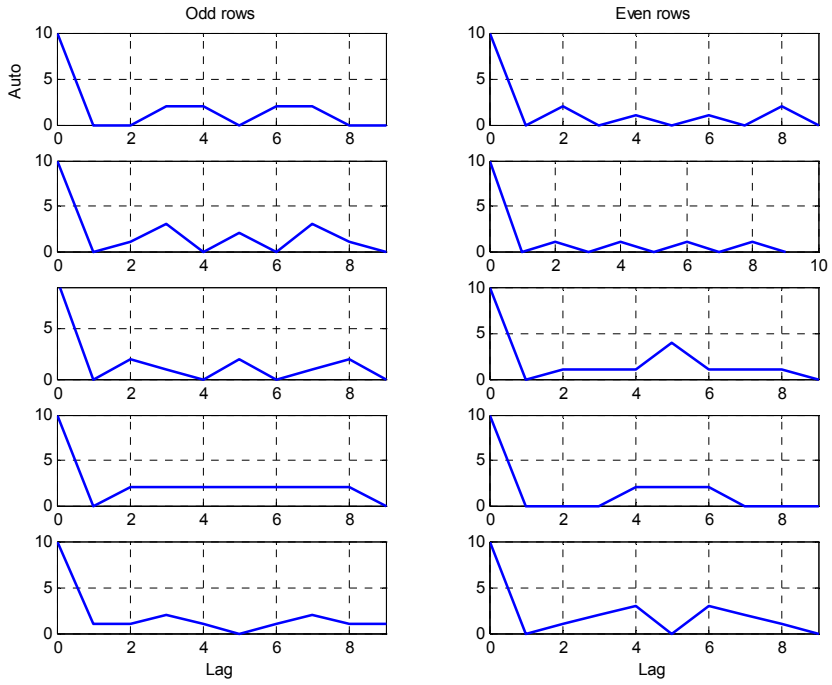


Figure A.6 Hamming autocorrelation functions for each sequence in the Markov code matrix shown in Figure A.2

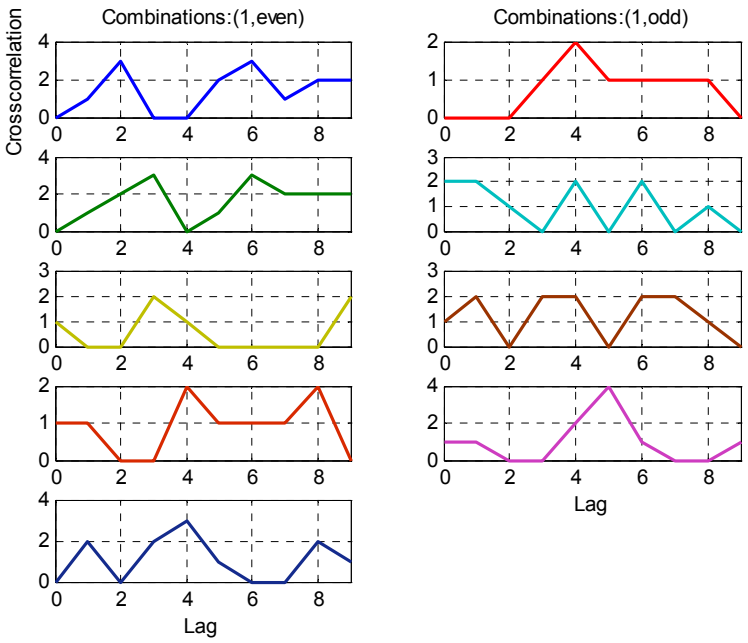


Figure A.7 Hamming cross-correlation functions for the combinations of row one with the rest from the set of Memoryless codes shown in Figure A.2

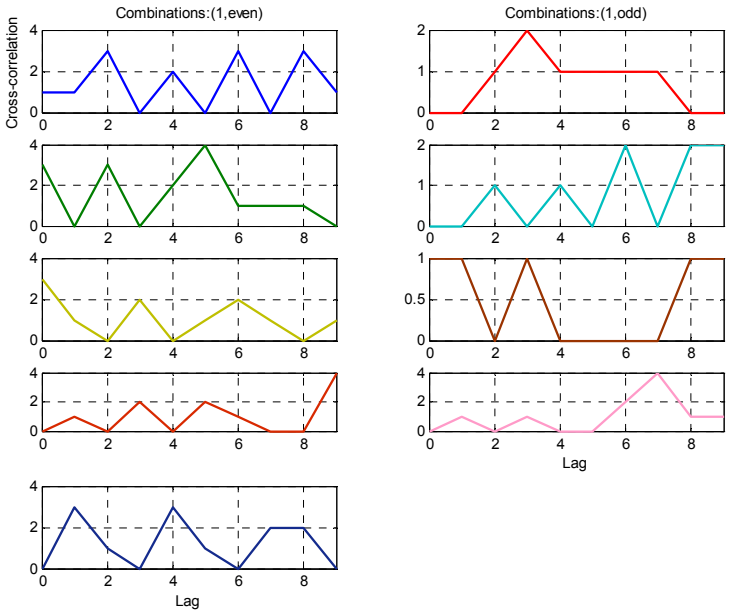


Figure A.8 Hamming cross-correlation functions for the combinations of row one with the rest from the set of Markov codes shown in Figure A.2

Figures A.7 and A.8 show the set of Hamming cross-correlation functions specifically for the first codeword in the Memoryless and Markov code sets shown in Figure A.2. If they

are examined, one can notice that no constant upper bound as in the case of CC code is possible to be found for all the cases. We performed the computations for the rest of the all possible distinct combinations in the set and the behaviour was found to be very similar.

All this non-uniformity observed in both functions for the case of the general random stationary process (i.e., Memoryless-based FH code) definitely contributed to the fact that the code set did not meet the Lempel-Greenberger optimality criterion cited before. Obviously, it is expected that a given CC FH code set behaves much better than a set of FH codes based on a general random stationary process for the same conditions. This *a priori* assumption is verified through some of the results shown in Chapter 4.