Adaptive Multi-Code Assignment for a DS-CDMA Ad Hoc Network

BRUNO LOW
(Diplôme d’ingénieur, INT)

Adaptive Multi-Code Assignment for a DS-CDMA Ad Hoc Network

BRUNO LOW
(Diplôme d'ingénieur, INT)

A Thesis submitted

For The degree of Master of Engineering

Department of Electrical and Computer Engineering
NATIONAL UNIVERSITY OF SINGAPORE

2005

Acknowledgements

I would like to thank my two supervisors Professors Marc Andre Armand and Mehul Motani for their invaluable advices. I would like also to acknowledge the great support from my family and friends Stephanie Yio and Moulay Rachid Elidrissi during these two years in Singapore. Finally, my appreciation to the contributions of Feng Cai and my two supervisors to the publication, "***Distributed Code Assignment for DS-CDMA Ad Hoc Network***" in the conference IEEE Digital Signal Processing and Digital Communication, Gold Coast (Australia) December 2003.

Table of Contents

List of Figures

List of Tables

List of Symbols and Annotations

AWGN	:	Additive White Gaussian Noise
BER	:	Bit Error Rate
BPSK	:	Binary Phase shift Keying
CAM	:	Code Assignment Message
CCA	:	Common Code Assignment
CCP	:	Code Correction Protocol
CD	:	Collision Detection
CSMA (-CD)	:	Carrier Sense Multiple Access
CDMA	:	Code Division Multiple Access
DS	:	Direct Sequence
FAMA	:	Floor Acquisition Multiple Access
FDMA	:	Frequency Division Multiple Access
INIT	:	Initialization Message
MAC	:	Multiple Access Control
MAI	:	Multiple Access Interference
PCAM	:	Pre-Code Assignment Message
PCAM_ACK	:	Pre-Code Assignment Message Acknowledgment
PN	:	Pseudo Noise
POCA	:	Pair wise Oriented Code Assignment

RandCA	:	Random Code Assignment
ROCA	:	Receiver Oriented Code Assignment
SINR	:	Signal to Interference plus Noise Ratio
SNR	:	Signal Noise Ratio
TDMA	:	Time Division Multiple Access
TOCA	:	Transmitter Oriented Code Assignment
$P_{k,l}$	:	Received power between the k^{th} transmitter and the l^{th} receiver
G_{ij}	:	Path loss expression between the k^{th} transmitter and the l^{th} receiver
$\widetilde{P_k}$	:	Transmission power of the k^{th} transmitter
γ_k	:	SINR threshold of the k^{th} receiver
T_b	:	Bit duration
T_c	:	Chip duration of the PN sequences
M	:	Length of the PN sequences
$s_{k,l}(t)$	:	Received signal from the k^{th} transmitter at the l^{th} receiver
$n(t)$	:	Additive White Gaussian Noise (AWGN) with variance $\frac{N_0}{2}$ and mean 0
D_{k,l,a_l}	:	Desired signal from the l^{th} transmitter at the k^{th} receiver matched to the PN sequence a_l
MAI_{k,l,a_l}	:	MAI at the k^{th} receiver matched to matched to the l^{th} transmitter using the PN sequence a_l
η	:	Thermal noise component

Φ_{i,l,a_l,k,a_k}	:	Interference created by the k^{th} transmitter using the PN sequence a_k at i^{th} receiver matched to the l^{th} transmitter using the PN sequence a_l
$C_{a_k,a_i}(l)$	:	Discrete aperiodic cross-correlation function between the PN sequences a_k and a_i with a chip delay l
$\theta_{a_k,a_i}(l)$	:	Even periodic cross-correlation function between the PN sequences a_k and a_i with a chip delay l
$\tilde{\theta}_{a_k,a_i}(l)$	:	Odd periodic cross-correlation function between the PN sequences a_k and a_i with a chip delay l
$\mu_{a_k,a_i}(l)$	:	Cross-correlation parameter between the PN sequences a_k and a_i with a chip delay l
r_{a_k,a_i}	:	Average cross-correlation parameter between the PN sequences a_k and a_i
$G(V,E)$	:	Graph representing the network
V	:	ensemble of nodes in the network
E	:	ensemble of edges in the network
Δ	:	Degree of the network
$M_x(i)$	:	Ensemble of x-hop neighbors of the node i
$L(l)$	:	Ensemble of adjacent links of the link l
C	:	Ensemble of PN sequences available for code assignment
N	:	Cardinal of the ensemble C

$CC(N \times N)$	:	Average cross-correlation matrix
$PN_i^{(T)}$	:	Set of PN sequences assigned for each transmitter with cardinal
$w_i^{(T)}$	:	Cardinal of $PN_i^{(T)}$
$PN_i^{(R)}$	:	Set of PN sequences assigned for each receiver
$w_i^{(R)}$	:	Cardinal of $PN_i^{(R)}$
$PN_i^{(P)}$	:	Set of PN sequences assigned for each link
$w_i^{(P)}$	:	Cardinal $PN_i^{(P)}$
Ψ_k	:	Code assignment vector of transmitter, receiver or transmitter-receiver pair k
$\Pi_{j,l}$	:	Collision cost vector associated to the j^{th} transmitter and the l^{th} receiver
Π_j	:	Collision cost vector associated to the j^{th} transmitter
T_j	:	Overall additional interference vector associated to the j^{th} transmitter

Abstract

Adaptive Multi-Code Assignment for a DS-CDMA Ad Hoc Network

by

Bruno LOW

Master of Engineering in Electrical and Computer Engineering

National University of Singapore (SINGAPORE)

Professor Marc Andre ARMAND and Professor Mehul MOTANI

DS-CDMA code assignments have been introduced as a solution for the hidden and exposed terminal problems for Ad Hoc Networks. Our works present new DS-CDMA multi-code assignment protocols and algorithms. The proposed multi-code assignment schemes are able to satisfy the receivers' bit rate, eliminate collisions and limit the effects of Multiple Access Interference (MAI). We introduce code assignment protocols for both centralized and distributed Ad Hoc Networks. We present analytical and simulation results for the proposed code assignments.

Keywords: Ad Hoc Network, DS-CDMA, Code Assignment, Collision, MAI

Summary

In this thesis we introduce different multi-code assignment algorithms for centralized and distributed Direct Sequence Code Division Multiple Access (DS-CDMA) Ad Hoc Networks. These algorithms are able to eliminate collisions and limit the effect of Multiple Access Interference (MAI).

Chapter 1 briefly introduces the existing wireless communication networks that we have today. Subsequently, the concept of ad hoc network and the issues to be considered in developing a reliable Multiple Access Control (MAC) protocol are given.

Chapter 2 shows how spread spectrum techniques can be used as modulation and multiple access tools for ad hoc networks. We present the multiple access problem in a DS-CDMA environment and give some important results. Subsequently, we show the different code assignment strategies used for a DS-CDMA ad hoc network to limit or eliminate collisions. We illustrate the mapping of the code assignment problems into graph coloring problems and highlight the importance of efficient code assignment protocol and algorithm during the correction process.

In Chapter 3, we introduce a code assignment scheme which can be used for a centralized or a small distributed network. We formulate the multi-code assignment problem and its

constraints using a Pairwise Oriented Code Assignment (POCA) strategy. Subsequently, a code assignment algorithm and a token passing protocol are presented for a small sized network. The scheme is able to control and correct the combined transmission power and the set of PN sequences assigned to each transmitter-receiver pair. We then propose a code diversity technique using the multiple code assignment to limit MAI. Thereafter, simulation results are presented. Finally, we conclude the chapter and draw the limitation of our system.

Chapter 4 introduces a new code assignment protocol and algorithm for a distributed Transmitter Oriented Code Assignment (TOCA) ad hoc network. This protocol must ensure that only one violating node amongst its 2-hop neighbors including itself corrects its PN sequences at a time. In the correction process, three different types of messages are exchanged among the nodes: the Code Assignment Message (CAM), the Pre-Code Assignment Message (PCAM) and the Pre-Code Assignment Message Acknowledgment (PCAM_ACK). The message pair PCAM-PCAM_ACK is used to lock the 1-hop neighbors of a violating node to ensure that the previous condition is met. We define a strategy to guarantee that at least the highest priority nodes amongst their 2-hop neighbors including themselves are correcting their PN sequences during the correction period. CAM messages are used for 1-hop and 2-hop code assignment updates. Subsequently, a new algorithm is presented which assigns to each transmitter the “best PN sequences” to limit collisions and MAI. We show the impact of the number of code sequences available, the transmission power and the mobility of nodes on the correction process.

Chapter 5 presents the implementation of the first real time simulation for a TOCA ad hoc network using the event-oriented simulator OMNeT++.

Chapter 6 concludes this thesis by summarizing the contributions made to the research community and by giving possible directions for future work.

Chapter 1 Introduction

In the first chapter, we give a brief introduction of modern wireless networks and key issues. Section 1.1 briefly gives the background of modern wireless communication systems. Section 1.2 describes the concept of ad hoc networks. Section 1.3 discusses the challenges involved in the design of Multiple Access Control (MAC) protocols in ad hoc networks. Finally, Section 1.4 presents the different contributions made for the research community as well as the organization of this thesis.

1.1. The Wireless Revolution

1.1.1. Cellular Networks

The wireless revolution started in the 1990s, when mobile phones were introduced in the market and changed our way of communicating with one another. Ever since, different cellular protocol standards such as Global System for Mobile Communication (GSM) and Code Division Multiple Access (CDMA) [RAP02] have appeared. The increasing popularity of mobile phones has surpassed that of traditional corded phones, and this number continues to escalate. New applications have since appeared, with the biggest success being the introduction of Short Message Service (SMS).

With the advancement in Internet technologies, customers' needs have evolved with more and more people wanting to stay connected through the Internet by way of emails and other messaging applications. Consequently, voice, video and data packets need to be carried over the cellular network. Further, different operators worldwide have been upgrading their networks from the 2nd to the 3rd generation to provide users with more possibilities by increasing the throughput of the network. With a much faster connection to the Internet and access to multimedia sources, mobiles phones today serve as personal computers, giving users the ability to watch videos, listen to music, enjoy video-conferences, and surf the Internet, in addition to making calls.

1.1.2. Non-Cellular Networks

While 3rd generation cellular networks have been delayed by excessive bandwidth frequency license and infrastructure costs, Wi-Fi also known as 802.11, uses a free bandwidth frequency and has a low infrastructure cost which provides broadband Internet access to users within a dozen to hundreds meters from designated Wi-Fi spots.

Wi-Fi technology has conquered the cities across the world such as Seoul, New York, Paris, and Singapore where the laptop or PDA users of the Wi-Fi networks receive broadband access to the Internet in waiting areas, business centers, a restaurant tabletop, hotel rooms or any other area within a venue in which Wi-Fi coverage is available.

1.2. Ad Hoc Networks

In [CHL03], a mobile ad hoc network is an autonomous system of mobile nodes connected by wireless links forming an arbitrary graph. Nodes are free to join, leave, move and organize themselves arbitrarily; thus, the mobile ad hoc network's topology may change rapidly and unpredictably. Its distributed structure obliges the nodes to be used as routers forwarding each other's messages to their final destinations. Such a network may be fully autonomous or connected to the larger Internet. It requires no prior investment in fixed infrastructure installation and little effort for its deployment.

There are a number of situations in which ad hoc networks are more adequate than installing fixed infrastructure. In sparse areas or for volatile networks, ad hoc networks have been proven to be cost effective. In disaster recovery, fixed infrastructure installation exists but cannot be depended on; thus, these networks are the only possible solution. Finally, at home or in the office, ad hoc networks which require no configuration are valid substitutes for local area networks.

Unfortunately, ad hoc networks are more difficult to implement than fixed networks. Nodes share the common wireless medium which distributed or random access protocols need to be defined resulting in poorer resources usage than centralized systems. Further, nodes' double capability as end-users and routers strongly increases the consumption of the limited power supply of the battery.

1.3. Multiple Access Control

In an ad hoc network, a node can be in the transmission range of multiple users or neighbors. In the case where users are sharing a single channel for transmission, if one neighbor is transmitting to a receiver and the others are kept idle at the same time, the packet will be received successfully, otherwise, if multiple neighbors are transmitting simultaneously to a receiver, packet collisions will occur. To completely eliminate the occurrence of packet collisions, nodes need to be aware of which of their two-hop neighbors are transmitting at any time. Unfortunately, such accurate information is impossible to obtain in an ad hoc network.

1.3.1. Hidden Terminal Problem

The hidden terminal problem results from the fact that two neighbors of a given node might not possibly hear each other. The problem occurs when the two nodes have a distance of two hops. Take for example Figure 1-1. The nodes *A* and *C* are neighbors of node *B,* but node *A* is not within the transmission range of node *C* and vice-versa. Therefore, when node *A* is transmitting, node *C* cannot sense it. Assuming now that node *C* starts transmitting simultaneously; thus the two packets from node *A* and node *C* will collide at node *B.* node *A* is said to be hidden from node *C.* Such a situation is a waste of channel capacity because the channel has been used for transmission but no packet has been received successfully.

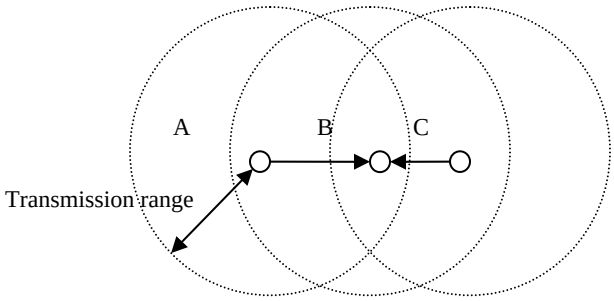


Figure 1-1 -The hidden terminal problem: A is "Hidden" from C

1.3.2. Exposed Terminal Problem

The second problem which might arise is "the exposed terminal problem". It results from the fact that a node which is transmitting may cause its neighbors to stay idle, even though the transmission from one of its neighbors may not disrupt the reception of the current transmitting packet. An example is given in Figure 1-2. Suppose that two nodes *B* and *C* are within each other's range, with node *B* transmitting to node *A*. node *C* wants to transmit to node *D*. The nodes *A* and *D* cannot hear the nodes *C* and *B* respectively. Node *C* will detect that the channel is busy and will not transmit even though transmission between the nodes *C* and *D* will not interfere with the communication between the nodes *B* and *A*. Node *C* is said to be exposed to node *B*. Contrary to the hidden terminal problem, in the exposed terminal problem, there is an underutilization of the channel.

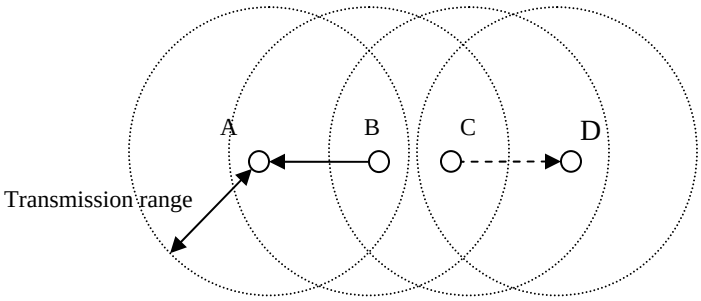


Figure 1-2-The exposed terminal problem: C is "exposed" to the node B

1.3.3. Random MAC Protocols

In a wireless communication system, multiple users are sharing the same medium for transmission; its control is managed via the Multiple Access Control (MAC) Protocol. ALOHA, the first MAC protocol for a packet radio network was introduced by N. Abramson [ABR70] in 1970. In ALOHA, packets are sent without prior channel sensing. The packet has a vulnerability period twice the size of the packet transmission time (a packet will not overlap another packet only if it is transmitted before or after the time duration of the packet). As a result the maximum throughput is very low (18% of the channel capacity). In Slotted-ALOHA [ABR73], nodes start transmitting packets only at the beginning of each time slot; thus, the vulnerability period is reduced to the size of the packet transmission time. As a result, Slotted-ALOHA compared to ALOHA, improves the maximum throughput but requires for nodes to be synchronized.

The Carrier Sense Multiple Access (CSMA) protocol [TOB75-1, TOB75-2] is a more sophisticated MAC protocol in which a node will first sense the channel and start transmitting only when the channel is idle. CSMA with collision detection (CSMA/CD) protocol improves CSMA by incorporating a special jamming signal at each receiver.

When a node is receiving two or more packets simultaneously, it will notify all users of the correction using this jamming signal. Retransmission schemes which improve channel utilization are the non-persistent, the 1-persistent and the p-persistent CSMA. However, the hidden and exposed terminal problems occur.

The Floor Acquisition Multiple Access (FAMA) is a generic term introduced by [FUL98] to describe channel acquisition strategies in order to eliminate the hidden and the exposed terminal problems by exchanging handshake packets or/and using busy tones. Before transmitting a packet, a node needs to acquire control of the channel in a manner that no collision will occur. However, the different FAMA algorithms which have been proposed involve a large exchange of header in the network which might considerably reduce the capacity of the channel.

1.3.4. Controlled MAC Protocols

In Frequency Division Multiple Access (FDMA) and Time Division Multiple Access (TDMA), the medium is respectively subdivided into non-overlapping frequency bands and timeslots. In FDMA, each node is transmitting through a narrow frequency band which results in greater vulnerability to multipath fading effects. In TDMA, every node must be synchronized otherwise different node's timeslot might overlap causing collisions.

Unlike FDMA and TDMA, Code Division Multiple Access (CDMA) [SCH77] divides the medium by using Pseudo Noise (PN) or code sequences. Each transmitter will

modulate its data signal using its assigned PN sequence and access the channel in a random manner. Different transmitting signals overlap both in time and frequency. A receiver demodulates the different incoming signals according to the assigned PN sequence. With users being able to simultaneously access the medium in a completely random manner in time and frequency, CDMA is a perfect candidate for use in ad hoc networks. Controlled MAC protocols aim to assign users available channels for transmission to entirely eliminate collisions. Such algorithms have been generalized in [RAM99].

1.4. Organization of the Thesis and Contributions

In this thesis, we will introduce, analyze and discuss new code assignment protocols and algorithms for Direct Sequence Code Division Multiple Access (DS-CDMA) mobile ad hoc networks.

Chapter 2 discusses the importance of spread spectrum techniques in ad hoc networks. We then present the multiple access problem in a DS-CDMA environment and give some important results. Subsequently, the different code assignment strategies proposed will be introduced. Finally, we will present in greater detail, the different protocols that have been developed in the past.

Chapter 3 presents our first solution to the multi-code assignment problem for a centralized or a small ad hoc network. The code assignment algorithm assigns to each transmitter-receiver pair a certain transmission power and a set of PN sequences to

minimize the total power consumption of the system and satisfy the bit rate requirement of each pair. At the receiver, MAI and the effects of fading are mitigated using a code diversity technique.

Chapter 4 introduces an adaptive multi-code assignment scheme at the transmitter end to ensure better response to the needs of different nodes. In this scheme, when a code violation is detected, the node moves into a correction phase to replace the violating PN sequences. A protocol has been developed to avoid the situation of having two nodes, which are two-hop apart, changing their PN sequences at the same time. Code selection is done via the code assignment algorithm which assigns to nodes, the "best PN sequences" from an initial set of PN sequences, in order to limit collisions and MAI in the network.

Chapter 5 presents in detail, the implementation of the adaptive multi-code assignment scheme proposed in Chapter 4 using the event simulator OMNeT++.

Finally, Chapter 6 summarizes the important contributions of this thesis and also discusses future work directions.

Chapter 2 Spread Spectrum and Ad Hoc Network

In this chapter, we begin with a discussion on the use of spread spectrum techniques as modulation and multiple access tools for mobile ad hoc networks. Spread spectrum communication was first introduced for military applications. Communications between transmitters and receivers are done by modulating data signals on a wideband carrier, and consequently the transmitted signal bandwidth is much larger than the data signal bandwidth. A spread spectrum-based CDMA system allows multiple transmitters to transmit simultaneously and each receiver to receive from many users. In such a system, multiple access results in random errors at the physical layer due to mutual interference [PUR77-1, PUR77-2, YAO77]. With efficient error control codes, for example, turbo code [PRO02], these errors may be efficiently corrected if the Signal to Interference plus Noise Ratio (SINR) is greater than a certain threshold. However, a spread spectrum system demands a more complex implementation for modulation and demodulation of the PN sequences.

Section 2.1 introduces the Direct Sequence Code Division Multiple Access (DS-CDMA) concept and highlights some important results. In Section 2.2, we present the code assignment problem for DS-CDMA ad hoc networks. The different strategies to solve

this problem are presented. Further, in Section 2.3, a discussion on related work will be conducted. Finally, Section 2.4 concludes this chapter.

2.1. DS-CDMA an Overview

2.1.1. Model and Assumption

In a spread spectrum-based DS-CDMA system using BPSK as a modulation technique, the received signal at the reference receiver 0 from the k^{th} transmitter amongst K users simultaneously transmitting is given by

$$s_{k,0}(t-\tau_k) = \sqrt{P_{k,0}}.a_k(t-\tau_k).b_k(t-\tau_k).\cos(\omega_c t+\phi_k) \tag{2.1}$$

where ω_c is the common center frequency, $a_k(t)$ is the PN sequence assigned to the k^{th} user, $b_k(t)$ is the data sequence of the k^{th} user, $P_{k,0}$ is the received signal after path loss of the k^{th} user at the reference receiver 0, ϕ_k is the carrier phase offset of the k^{th} user relative to a reference transmitter 1, τ_k is the delay of the k^{th} user relative to a reference transmitter 1.

$a_k(t)$ and $b_k(t)$ are both binary sequences with values from the set $\{+1,-1\}$. The PN sequence $a_k(t)$ can be written

$$a_k(t) = \sum_{i=-\infty}^{+\infty} a_{k,i}\, p_{T_c}(t - iT_c) \qquad a_{k,i} \in \{+1, -1\} \tag{2.2}$$

where T_c is the chip duration of the PN sequence. $p_T(t)$ is the unit pulse function of width *T*, and is defined by

$$p_T(t) = \begin{cases} 1 & \text{if } t \in [0, T] \\ 0 & \text{elsewhere} \end{cases} \tag{2.3}$$

Further, the $a_k(t)$ are periodic sequences of period $MT_c = T_b$ and satisfy

$$a_{k,M.l+j} = a_{k,j} \qquad \forall l \in \mathbb{Z} \tag{2.4}$$

where M is the number of chips of the PN sequences and T_b is the bit period when the PN sequences $a_k(t)$ are repeated. The data sequence $b_k(t)$ is given by

$$b_k(t) = \sum_{i=-\infty}^{+\infty} b_{k,i}\, p_{T_b}(t - iT_b) \qquad b_{k,i} \in \{+1, -1\} \tag{2.5}$$

Figure 2-1 shows the block diagram of an asynchronous multiple access DS-CDMA system where the reference receiver 0 is synchronized with the 1st transmitter, so that we can write $\tau_1 = 0$ and $\phi_1 = 0$.

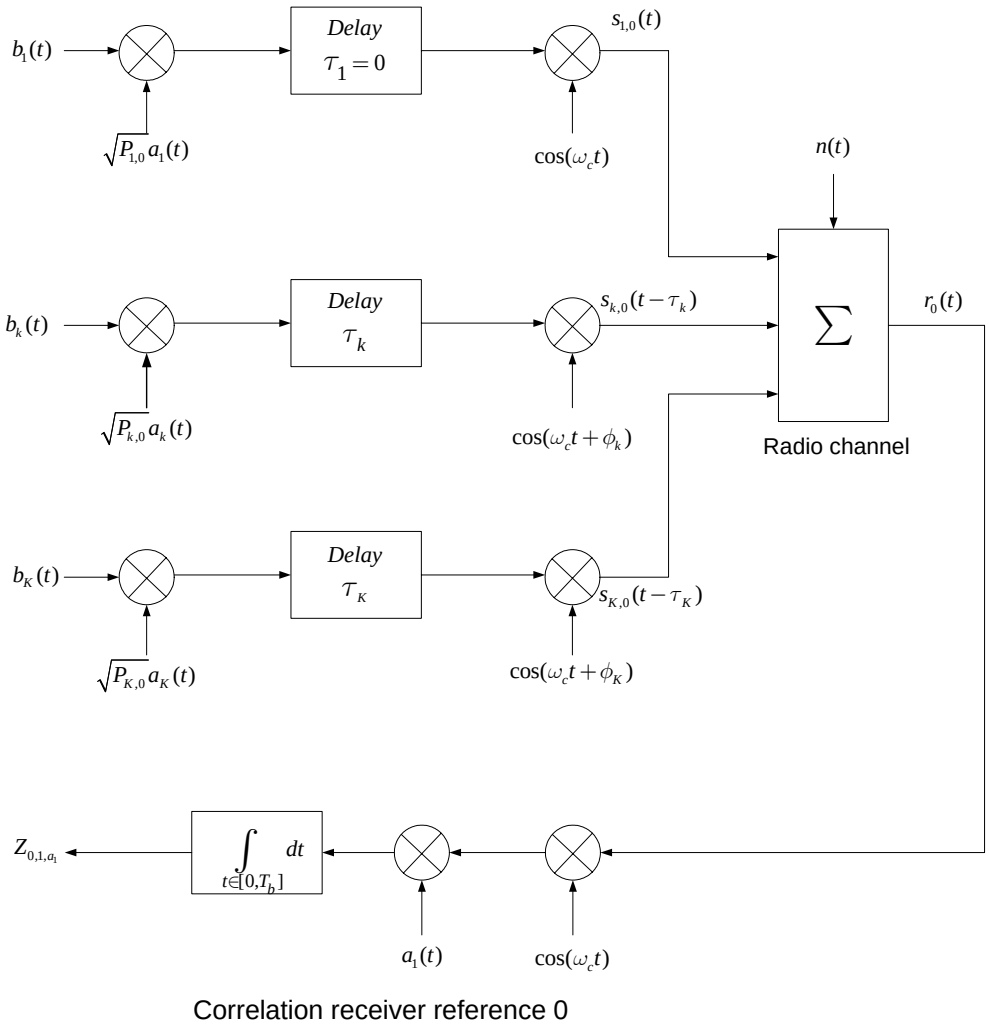


Figure 2-1 - Block diagram for a DS-CDMA multiple access system model in an AWGN channel

From Figure 2-1, the received signal at the input of the correlation receiver 0 can be expressed as

$$r_0(t) = \sum_{k=1}^{K} s_{k,0}(t-\tau_k) + n(t) \tag{2.6}$$

where $n(t)$ is the additive white Gaussian noise with variance $\frac{N_0}{2}$ and zero mean.

A more detailed description can be found in [SCH77].

2.1.2. Average SINR for an Asynchronous DS-CDMA System

Michael Pursley [PUR77-1, PUR77-2] conducted in-depth analysis of multiple access communication using spread spectrum technique. In particular, he closed-form derived expressions for the average Signal to Interference plus Noise Ratio (SINR) and the Bit Error Rate (BER) for an asynchronous DS-CDMA system. Further, in [YAO77], some analyses have been done for code design based on the average cross-correlation between the PN sequences.

In [PUR77-1], the average $SINR_{0,1,a_1}$ corresponding to the signal transmitted by the 1st user matched filtered by the receiver 0 using the PN sequence $a_1(t)$ is given by

$$SINR_{0,1,a_1} = \frac{P_{1,0}}{\frac{1}{6M^3}\sum_{k=2}^{K} P_{k,0}.r_{a_k,a_1} + \frac{N_0}{T_b}} \tag{2.7}$$

where r_{a_k,a_1} is the mean cross-correlation between the PN sequences $a_k(t)$ and $a_1(t)$ used by the k^{th} and the 1st transmitters respectively. The full derivation of (2.7) is given in Appendix A.

2.1.3. Cross-Correlation Parameters

In the previous section, we have given a simple expression for the average SINR for an asynchronous CDMA system. As can be seen, the average SINR depends on the r_{a_k,a_1}. The average r_{a_k,a_1} give indication of the quality of the PN sequences family used. In [KAR92], numerical evaluation of the cross correlation parameters presented in Appendix A is given and summarized in table 2-1.

Table 2-1- Cross-correlation parameters for CDMA code families adapted from [KAR92]

Family	M	r_{a_k,a_l}	$2\mu_{a_k,a_l}(0)$	$\mu_{a_k,a_l}(1)$	$2M^2$	$\frac{\left\|r_{a_k,a_l}-2M^2\right\|}{r_{a_k,a_l}}\times 100\%$
Gold	*31*	1674	1926	-256	1922	15%
m-sequence	*31*	1742	1918	-176	1922	10%
Gold	63	7398	8286	-888	7938	7.3%
Kasami(S)	63	6982	6958	24	7938	14%
Kasami(L)	63	8686	7982	704	7938	8.6%
m-sequence	63	8006	8110	-104	7938	0.8%
Gold	127	33682	32742	940	32258	4.2%
m-sequence	127	30902	31406	-504	32258	4.4%
Kasami(S)	255	138310	132430	5880	130050	6.0%
Kasami(L)	255	132322	138246	-5924	130050	1.7%
m-sequence	255	132206	132830	-624	130050	1.6%
Gold	511	536642	518198	18444	522242	2.7%
m-sequence	511	487430	488718	-1288	522242	7.1%
Gold	1023	2093414	2125342	-31928	2093058	0.02%
Kasami(S)	1023	2094766	2116766	-22000	2093058	0.08%
Kasami(L)	1023	2154770	2093302	61468	2093058	2.9%
m-sequence	1023	2059262	2049982	9280	2093058	1.6%
Gold	2047	8175130	8140742	34388	8380418	2.5%
m-sequence	2047	8460494	8460494	42160	8380418	0.9%
Kasami(S)	4095	32789578	32789578	-567852	33538050	2.3%
Kasami(L)	4095	34645122	34645122	745292	33538050	3.2%
m-sequence	4095	33725918	33725918	115744	33538050	0.6%

2.2. Code Assignment in Ad Hoc Network

2.2.1. Strategies for Code Assignment

DS-CDMA code assignment schemes require that nodes are transmitter, receiver or transmitter-receiver agile. A node is said to be transmitter, receiver, or transmitter-receiver agile when it is able to transmit, receive, or transmit and receive over a multitude of PN sequences, respectively. Proper code assignment schemes are needed to eliminate hidden terminal problems. Different strategies will be analyzed and discussed in detail in the following sections.

2.2.1.1. CCA

The simplest code assignment protocol, the Common Code Assignment (CCA), uses a common spreading code for transmission and reception of all packets. Collisions are reduced in CCA-ALOHA compared to non-spreading ALOHA. If two transmitted packets are overlapping in time and received in an asynchronous mode, they might not collide. The receiver will be synchronized with the first arriving packet. The second packet will have the same effect at the receiver as MAI and will not be received. In this way, the number of collisions is reduced. In addition, the system has a simple implementation [ABR94].

2.2.1.2. RandCA

In [PUR87, ABR94], the Random Code Assignment (RandCA) scheme uses a bank of spreading codes for transmission. A node will select a PN sequence randomly from the PN sequence bank to transmit packets, and at the intended receiver, all the packets received are demodulated using the same PN sequence. RandCA does not require any control protocol; however, the design of the receiver has a complexity which increases with the size of the bank. Usually the random code assignment is coupled with a channel access protocol. The most commonly used protocols are the slotted or non-slotted ALOHA.

2.2.1.3. ROCA

The Receiver Oriented Code Assignment (ROCA) scheme of [SOU88] assigns for each receiver, a PN sequence. When a packet needs to be sent, the node tunes its PN sequence for transmission to the desired receiver's PN sequence assigned by the ROCA scheme. The transceivers are said to be transmitter agile. Unfortunately, collisions can only be reduced but not completely eliminated; when two nodes are transmitting to the same receiver simultaneously, collisions occur. However, its hardware implementation is simple (because each node only has to listen to the nodes transmitting over its assigned PN sequences) and therefore less costly.

2.2.1.4. TOCA

The Transmitter Oriented Code Assignment (TOCA) scheme introduced by Makansi [MAK87], assigns a PN sequence for each transmitter. The transceiver is said to be receiver agile. Multiple users can transmit simultaneously to the same receiver which tunes its PN sequences for reception to different transmitters. The TOCA scheme has the advantage of authorizing multiple user communication and can entirely eliminate collisions. But, it requires more complex implementation than the ROCA scheme.

2.2.1.5. POCA

The Pairwise-Oriented Code Assignment (POCA) scheme of [HU93] assigns a PN sequence to each transmitter-receiver pair or link. For communication to be established, both transmitter and receiver have to tune their PN sequence for transmission and reception to the PN sequence assigned for the pair. The transceiver is said to be receiver-transmitter agile. POCA has the same capability as TOCA to eliminate collisions, but its implementation is much more complex and therefore much more costly.

2.2.2. Graph Coloring Problem

The code assignment problem can be mapped to a graph coloring problem. The network can be described as a graph $G(V,E)$, where V is the ensemble of vertices or nodes and E is the ensemble of edges or links. An element $e \in E$, can also be defined as $e = \{s,d\}$, where $s,d \in V$. The bank of PN sequences available is defined by $C = \left\{c_j\right\}_{j=1}^{N}$ where N

is the cardinality of *C*. We define the ensemble of *x*-hop neighbors of the node *u* (*y* is a x-hop neighbor of *u*, if *y* can reach *u* through x distinct hops) as

$$M_{\mathrm{x}}(u) = \{ y \in V : y \text{ is a x-hop neighbor of } u \} \tag{2.8}$$

and the ensemble of adjacent links of the link *l* as

$$L(l) = \{ e \in L : e \cap l = \varnothing \} \tag{2.9}$$

2.2.2.1. ROCA and TOCA Multi-Code Mapping

In TOCA and ROCA schemes, the number of collisions is minimized by assigning distinct pairs of two-hop neighbors, distinct PN sequences. Therefore, TOCA and ROCA problems can be reduced to the same graph coloring problem. The TOCA problem is equivalent to find subsets $PN_i^{(T)} \subset C$ such that

$$\begin{cases} \left| PN_i^{(T)} \right| = w_i^{(T)} & \forall i \in V \\ PN_i^{(T)} \cap PN_j^{(T)} = \varnothing & \forall i \in V, \forall j \in M_2(i) \end{cases} \tag{2.10}$$

where $w_i^{(T)}$ is the number of PN sequences required by the i^{th} transmitter and $PN_i^{(T)}$ is the set of PN sequences assigned to the i^{th} transmitter.

Similarly, ROCA problem is to find subsets $PN_i^{(R)} \subset C$ such that

$$\begin{cases} \left| PN_i^{(R)} \right| = w_i^{(R)} & \forall i \in V \\ PN_i^{(R)} \cap PN_j^{(R)} = \varnothing & \forall i \in V, \forall j \in M_2(i) \end{cases} \tag{2.11}$$

where $w_i^{(R)}$ is the number of PN sequences required by the i^{th} receiver and $PN_i^{(R)}$ is the set of PN sequences assigned to the i^{th} receiver. Figure 2-2 depicts a TOCA and ROCA single code assignment for a given graph.

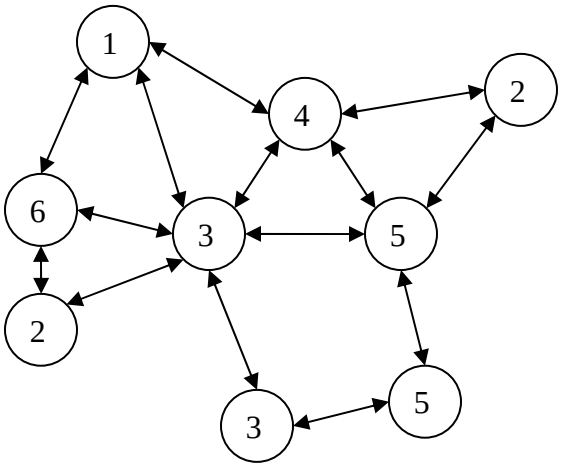


Figure 2-2-TOCA and ROCA single code assignment using 6 PN sequences

2.2.2.2. POCA Multi-Code Mapping

In the POCA scheme, the hidden terminal problem is totally eliminated by assigning distinct pairs of adjacent transmitter-receiver links, distinct PN sequences. The POCA problem is equivalent to find subsets $PN_l^{(P)} \subset C$ such that

$$\begin{cases} \left|PN_l^{(P)}\right| = w_l^{(P)} & \forall l \in E \\ PN_l^{(P)} \cap PN_e^{(P)} = \varnothing & \forall l \in E, \forall e \in L(l) \end{cases} \tag{2.12}$$

where $w_l^{(P)}$ is the number of PN sequences required by the l^{th} link, and $PN_l^{(P)}$ is the set of PN sequences assigned to the l^{th} link. Figure 2-3 depicts a POCA single code assignment for an arbitrary graph.

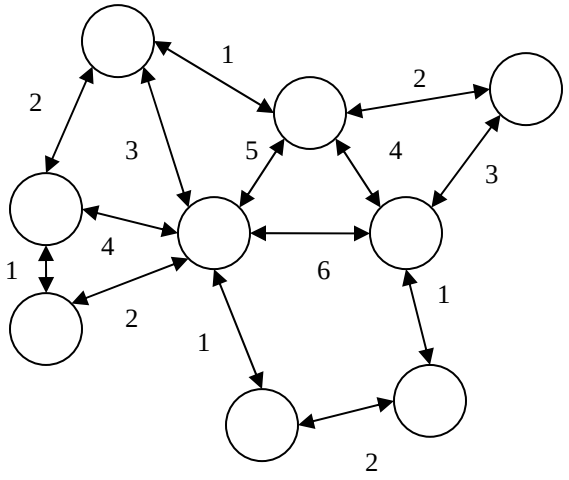


Figure 2-3 - POCA single code assignment using 6 PN sequences

2.2.2.3. Graph Coloring Bounds

[HU93, MAK87] have given lower and upper bound on the number of PN sequences needed for the single code assignment problem in order to completely eliminate packet collisions. In [BAT99], the number of PN sequences needed for a violation-free TOCA network has been studied through simulation. A violation occurs when the conditions (2.10), (2.11) or (2.12) are not satisfied for the TOCA, ROCA or POCA schemes, respectively.

Theorem 1*: For a violation-free TOCA or ROCA network, the minimal number of PN sequences needed must be at least equal or greater than the maximal degree* Δ *of the network.*

Proof: Any node has at most Δ neighbors. Each distinct neighbor must be assigned a distinct PN sequence to satisfy the condition (2.10) or (2.11) and the Theorem 1 follows.

Theorem 2*: if* $N \geq \Delta(\Delta-1)+1$ *and each user is assigned one PN sequence, a violation-free TOCA or ROCA network can be guaranteed.*

Proof: Any node has at most $\Delta(\Delta-1)$ two-hop neighbors. Due to the constraints given in (2.10) and (2.11) for TOCA and ROCA schemes, any node must be assigned a distinct PN sequence from its two-hop neighbors. If each 2-hop neighbor uses a unique PN sequence and the number of PN sequences available is greater than $\Delta(\Delta-1)$, then the node at hand will always have at least one PN sequence available to choose from *C* and the Theorem 2 follows.

Theorem 3*: For a POCA network, the minimal number of code sequences needed to eliminate the hidden terminal problem has to be at least* Δ.

Proof: All the edges emanating from or terminating at any node, must be assigned distinct PN sequences to satisfy the POCA constraint in (2.12) and the Theorem 3 follows.

Theorem 4*: if* $N \geq \Delta + 1$ *and each link is assigned one PN sequence, a violation-free POCA network can be guaranteed.*

The proof for Theorem 4 can be found in [MIS92].

In [MIS92], a heuristic approach is used to find the perfect solution using $\Delta + 1$ PN sequences. However, the construction of the solution is time consuming with high complexity. The frequent topological changes imply excessive number of re-computation for code assignments among the nodes within the network; thus, the algorithm [MIS92, HU93] can only be implemented for a centralized or a static network.

Hu [HU93] has implemented a fast code assignment algorithm for POCA-based networks for the number of PN sequences $N \geq 2(\Delta - 1) + 1$. This number is more suitable for the code correction process, in which one link can correct its code sequence without imposing a change for other links as Theorem 4 implies.

Theorem 5*: If* $N \geq 2\Delta - 1$ *and each link is assigned one PN sequence, a violated link can correct its PN sequence without imposing a change for other links .*

Proof: Any link has at most $2(\Delta - 1)$ adjacent links. Due to the constraints given in (2.12) for the POCA scheme, any link must be assigned a distinct PN sequence from its adjacent links. If each adjacent link uses a unique PN sequence and the number of PN

sequences available is greater than $2(\Delta-1)$, then the link at hand will always have at least one PN sequence available to choose from *C* and the theorem 5 follows.

2.3. Related Work

2.3.1. Centralized Algorithms

A centralized algorithm assumes the existence of a super node which has the knowledge of the network's topology. The super node can be selected by leader election [NAK00, NAK02] and is informed by other nodes of any change in the topology. The super node watches the network and corrects PN sequences when violations are sensed. Thus, centralized algorithms are only practically viable in small networks with relatively slow motion. In such cases, centralized code assignment algorithms produce better performances than distributed versions.

In [MAK87], the procedure for code assignment in an arbitrary network is given by: 1) Choose randomly a node in the network, and assign it the code sequence c_1. 2) Choose randomly an unassigned node, and try to assign it the code sequence c_1; if any of its two-hop neighbors have already been assigned the code sequence c_1, then try with the code sequence c_2. Continue until a valid code is found. 3) Repeat until all the nodes have a code sequence.

Makansi [MAK87] has also proposed a perfect code assignment scheme for networks with special topologies using the minimal number of code sequences Δ. The algorithm extracts a 2-hop cell for which its replication can generate the complete topology of the network. Subsequently, it assigns a code sequence for each element of the 2-hop cell using Δ code sequences ensuring that no code violation occurs after the cell replication process to construct the complete network. This code assignment algorithm guarantees a violation-free TOCA network using Δ code sequences for special topologies such as the bus, the hexagonal and the grid networks.

In [MAK87], the TOCA scheme and a channel access protocol are used jointly. Before transmitting a data packet, the pair messages Ready-to-Send (RTS) and Clear-to-Send (CTS) are exchanged between the transmitter and the receiver to verify that the receiver is idle.

In [HUN92], Hung proposed a code assignment algorithm which requires less PN sequences than the Dsatur algorithm. The algorithm ranks the nodes in order of decreasing degrees, it first assigns the node with the maximum degree with the code sequence c_1 and its neighbors with the code sequences c_2 to $c_{\Delta+1}$. Subsequently, it assigns a code sequence to the other ranked nodes, in order to minimize the number of code sequences currently used and the binding function of each node which represents the degree of freedom left for the code assignment of its remaining 1-hop and 2-hop neighbors.

In [HU93], the TOCA algorithm presented differs from [MAK87] by fixing the number of PN sequences available to $\Delta(\Delta-1)+1$ where Δ is the maximal degree of the network. The algorithm follows these 4 steps: 1) Select any node without a code sequence, 2) Remove from the set of code sequences available the sequences used by its two-hop neighbors 3) Select randomly a code sequence from this set. 4) Continue until all nodes are assigned a code sequence.

[HU93] introduced two POCA algorithms using $\Delta+1$ and $2(\Delta-1)+1$ code sequences. The first algorithm using $\Delta+1$ code sequences is a very complex heuristic involving multiple code sequence permutations among links. This algorithm is detailed in [MIS92]. The second algorithm using $2(\Delta-1)+1$ code sequences does not require any code sequence permutation among links. The algorithm is described as follows: 1) Select any link without a code sequence. 2) Remove from the set of code sequences available the sequences used by its adjacent links. 3) Select randomly a code sequence from this set. 4) Continue until all links are assigned a code sequence.

Wu [WU02] proposed a combined code assignment and power control scheme for POCA networks. This scheme aims to minimize the total power consumption or the congestion level of the network.

2.3.2. Distributed Algorithms

[HU93] derived algorithms for TOCA initialization and correction processes. The TOCA initialization process is used to initialize quickly the network by assigning a code sequence to each node. One of his algorithms is the sink tree algorithm which will be described and discussed in Section 4.2.3. The TOCA correction process is used to correct the PN sequences of violating nodes. [HU93] introduced two protocols for the code correction process: the token-passing and the deadlock-free orientation protocols. The two protocols are used to avoid that a pair of two-hop neighbors are correcting their PN sequence simultaneously in which another violation might occur. This experience is called deadlock. Using the token passing algorithm, a node can only change its own code sequence when it receives a token packet containing enough information for the node to select a new code sequence. The deadlock orientation protocol specifies the correcting chain of potentially conflicting nodes in such a way that no deadlock can occur. The code assignment used is similar to the centralized algorithm previously introduced by [HU93].

[BER95] introduced another distributed TOCA algorithm which minimizes the number of PN sequences used. The algorithm follows the 3 steps: 1) each node assigns to itself the lowest indexed code sequence which is not used by any of its 2-hop neighbors with lower identification (ID). It then informs its 1-hop and 2-hop neighbors of its code assignment by sending a Code Assignment Message (CAM). 2) If a CAM packet is received from a 2-hop neighbor with a lower ID sharing the same code sequence, the node will subsequently correct its code sequence following the 1st step. 3) The code

assignment algorithm is completed when the node and its 2-hop neighbors with lower ID are assigned a code sequence.

Such code assignment scheme is not suitable for mobile ad hoc networks due to its high complexity, where frequent topological changes occur.

[GAR97] introduced a combined code assignment and channel access protocol scheme. In this scheme, the transmitter and receiver exchanges the pair RTS-CTS packets on the control channel before transmitting a burst packet. The code assignment of the transmitter is set in the RTS. Upon its reception, the idle intended receiver generates the CTS packet and tunes its correlation receiver to the code sequence for the burst packet reception. Nonetheless, by using a handshaking with TOCA or POCA, the system looses its multi-user capability to receive multiple packets from non-overlapping code sequences.

2.4. Conclusion

In this chapter, we gave an overview of the DS-CDMA system. We then introduced and discussed the existing code assignment strategies which limit collisions for ad hoc networks. We have shown that only POCA and TOCA can completely overcome the hidden and exposed terminal problems. In related works, the focus has been on a single code assignment model. Also, mobility in ad hoc networks has most of the time been omitted, and therefore, a correction process when code violation occurs has almost never been discussed. Only [HU93, GAR97] pointed out the problem of graph re-coloring and designed protocols to correct code sequence violations.

Chapter 3 Multi-Code Assignment for Small POCA Networks

In this chapter, we propose a novel DS-CDMA code assignment scheme for small POCA networks. Multi-PN sequences are assigned for each transmitter-receiver pair to satisfy its bit rate requirement. Code assignment and power control are combined to eliminate collisions, to mitigate the effects of MAI and fading and to minimize the overall power consumption in the ad hoc network.

3.1. Model and Assumption

3.1.1. The DS-CDMA Model

We consider a DS-CDMA ad hoc network system; the network can be represented as a collection of *K* transmitter-receiver pairs which communicate simultaneously [WU02]. We denote by $\partial_{j,k}$, the shortest path in number of hops between the node *j* and *k*, and by h_j, $h_j = \max\{\partial_{j,k} : k \in V\}$. The 'leader node' [LIN97][NAK02] is chosen as the node which minimizes the h_j. $h_{\min}$ is defined by

$$h_{\min} = \min_{j \in V}(h_j) \tag{3.1}$$

where V is the ensemble of nodes in the ad hoc network. If $h_{\min} \leq 3$, pseudo-synchronization can be done amongst the nodes in the ad hoc network [ROM01].

For the entire system, the set of PN sequences $C = \{c_j\}_{j=1}^{N}$ is available for the code assignment where the normalized PN sequences are defined by

$$c_j = \frac{1}{\sqrt{M}} \sum_{i=1}^{M} c_{j,i}\, p_{Tc}(t - iT_c) \qquad \text{with } c_{j,i} = \{-1; +1\} \qquad \forall j \leq N \tag{3.2}$$

where M is the number of chips of the PN sequence, $p_T(t)$ the pulse function of parameter T and T_c the chip duration.

In the pseudo-synchronization condition, the code sequences are assumed to be orthogonal and are normalized, the PN sequences satisfy the condition

$$\langle c_j, c_i \rangle = \delta_{ij} \qquad \forall\, j, i \leq N \tag{3.3}$$

where $\langle\ .\ ,\ .\ \rangle$ represents the scalar product and δ_{ij} is defined by

$$\delta_{ij} = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \tag{3.4}$$

Each k^{th} transmitter-receiver pair is assigned the subset $PN_k^{(P)}$ of cardinal $w_k^{(P)}$ from *C*. For better clarity, the subset $PN_k^{(P)}$ is simplified to PN_k and its cardinal to w_k. We introduced the code assignment vector $\Psi_k = \{\varphi_{ki}\}_{i=1}^{N}$ associated with the k^{th} transmitter-receiver pair and its elements are given by

$$\varphi_{k,i} = \begin{cases} 1 & \text{if } c_i \in PN_k \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

The transmission power of the k^{th} transmitter is denoted by $\widetilde{P_k}$, and the path loss between the k^{th} transmitter and the l^{th} receiver, is represented by G_{kl}. A method to obtain the G_{kl} will be introduced in the following section. Finally, the variance of the additive white Gaussian noise for the k^{th} channel is denoted by σ_k^2. The k^{th} correlation receiver is comprised of w_k matched filters; each of them is matched to a distinct code sequence $pn_{k,i}$. The received signals are synchronized by each matched filter. It is assumed that each of the w_k channels are weighted with the same weight $\frac{1}{w_k}$. Thus, we can express the signal to interference plus noise ratio $SINR_k$ associated to the k^{th} transmitter–receiver pair as

$$SINR_k = \frac{\widetilde{P_k}.G_{kk}}{\frac{1}{w_k}\sum_{\substack{j\le K \\ j\neq k}} \widetilde{P_j}.G_{jk}.\left(\Psi_j.\Psi_k{}^T\right) + \sigma_k^2} \tag{3.6}$$

where A^T is the transposition of the vector A.

3.1.2. Formulation of the Power Control Problem

In an ad hoc network, the power consumption of the whole system has to be minimized according to the SINR constraint of each k^{th} receiver $SINR_k \geq \gamma_k$, where γ_k represents the SINR threshold of the k^{th} pair. The power control problem is to determine the best subset PN_k and transmission power $\widetilde{P_k}$ in order to minimize the total power consumption while satisfying the SINR requirement, i.e.

$$\text{minimize} \sum_{k \leq K} \widetilde{P_k} \quad \text{such that} \quad SINR_k \geq \gamma_k \ \forall k \leq K \tag{3.7}$$

[WU02] expressed the condition in (3.7), using the $K \times K$ matrix F and the vectors of dimension K, u and $\widetilde{P}$ by

$$(I - F).\widetilde{P} \geq u \tag{3.8}$$

where the elements F_{kj} of F are defined by $F_{kj} = \dfrac{G_{jk}\gamma_k\left(\Psi_j.\Psi_k{}^T\right)}{G_{kk}w_k}$, the elements u_j of u by $u_j = \dfrac{\gamma_j\sigma_j^2}{G_{jj}}$, $\widetilde{P}$ is the transmission power vector of elements $\widetilde{P_j}$, and I denotes the $K \times K$ identity matrix.

From (3.8) and assuming that the matrix $I-F$ has an inverse, the optimal vector power $\widetilde{P_{opt}}$ is given by

$$\widetilde{P_{opt}} = (I-F)^{-1}u \tag{3.9}$$

In our simulations, $(I-F)^{-1}$ always exists, and thus justifies this assumption.

3.2. Minimizing the Power Consumption

Using the Taylor series expansion for equation (3.9), we can derive

$$\sum_{k=1}^{K}\widetilde{P_k} = \left\|(I-F)^{-1}u\right\|_1 \approx \left\|(I+F)\ u\right\|_1 = \left\|u\right\|_1 + \left\|Fu\right\|_1 \tag{3.10}$$

where $\|\ .\ \|_1$ denotes the 1-norm, i.e. the sum of the absolute value of the elements of the vector. This approximation is true only if the maximum eigenvalue of *F* is much smaller than 1. Then, minimizing $\sum_{k=1}^{K}\widetilde{P_k}$ is equivalent to solve the weighted total squared cross-correlation (WTSC) minimization problem [WU02] by minimizing $\left\|Fu\right\|_1$. The WTSC problem is given by

$$\text{minimize } \|Fu\|_1 \Leftrightarrow \text{minimize} \sum_{k \le K} \sum_{\substack{j \le K \\ j \ne k}} \nu_{jk} \left(\Psi_j . \Psi_k{}^T \right) \tag{3.11}$$

where $\nu_{jk} = \dfrac{G_{jk} \gamma_k \gamma_j \sigma_j^2}{G_{kk} w_k G_{jj}}$.

3.3. Code Assignment Algorithms

We define the subset C' of the set C of PN sequences available, as the ensemble of code sequences currently not being used in the network. The subset E' of the ensemble E of transmitter-receiver pairs in the network represents the ensemble of links which have already been assigned code sequences. By analyzing the WTSC problem, we see that if $j \ne k$, the k^{th} transmitter corrupts the communication between the j^{th} pair, and conversely, the communication between the k^{th} pair is also corrupted by the j^{th} transmitter. The total interference between these two transmitter-receiver pairs can be represented by the sum $\nu_{jk} + \nu_{kj}$.

3.3.1. Code Initialization

The leader node initializes the code assignment for each transmitter-receiver pair using the code initialization algorithm. In this section, the knowledge of the ν_{jk} by the leader node is assumed. The leader initializes $C' = C$ and $E' = \varnothing$. The code initialization algorithm is a heuristic given by

1. Sort the transmitter-receiver pairs in ascending order according to the measure $\sum_{\substack{j\le K \\ j\neq k}} \nu_{jk} + \nu_{kj}$, and let *u(1), u(2), …, u(K)* be an enumeration of pairs with respect to this order.
2. For each *k* from 1 to *K*,

   If (C' is not empty)

   The code sequences of the set $PN_{u(k)}$ are chosen randomly from C'. Then, set $C' = C' - PN_{u(k)}$ and $E' = E' \cup \{u(k)\}$.

   else

   Compute the total interference vector $I_{u(k)}$ defined in (3.12) and extract the set $PN_{u(k)}$ from C satisfying (3.13). Then, set $E' = E' \cup \{u(k)\}$.
3. When the entire system has satisfied their required number of distinct PN sequences, we extracted the transmission power vector $\widetilde{P_{opt}}$ from (3.9) .

We define the total interference vector $I_{u(k)}$ of dimension N associated with the $u(k)^{th}$ link which each of its elements $I_{u(k),l}$ represents the possible total interference between the $u(k)^{th}$ transmitter-receiver pair and the other pairs of *E'* by choosing the PN sequence c_l. These elements $I_{u(k),l}$ are defined by

$$I_{u(k),l} = \sum_{j \in L \cap c_l \in PN_j} \nu_{j,u(k)} + \nu_{u(k),j} \tag{3.12}$$

The PN sequences chosen minimize the possible total interference between the *u(k)*th link and the other links of *E'*. $PN_{u(k)}$ satisfies the condition given by

$$\sum_{c_l \in PN_{u(k)}} I_{u(k),l} \leq \sum_{c_l \in X} I_{u(k),l} \quad \text{such that } |X| = \left|PN_{u(k)}\right| \ \forall X \subset C \tag{3.13}$$

Practically, the PN sequences corresponding to the lowest elements of the vector $I_{u(k)}$ are chosen.

3.3.2. Code Correction

The code correction process is used to re-adjust the code assignment and the transmission power of each link. This process is executed by the transmitter-receiver link itself. We suppose that the ν_{jk} are known by each transmitter-receiver pair. The code correction algorithm is reduced to the steps 2 and 3 of the initialization process.

3.4. The Token Circulation

In this section, we describe how a token ring may be used to disseminate topology information to the transmitter-receiver pairs in the network, to detect new users in the network, and to schedule the PN sequence updates for each pair.

3.4.1. Assumptions and Definitions

We assume that the graph constituting the receivers is connected, i.e., a path exists between every pair of receivers so that if one receiver leaves or joins the network, the graph will still be connected. Furthermore, every receiver is able to sense a signal from any sender.

We assume that a network layer routing exists, and that all nodes are able to update their neighbor table. By using this table, a sender will be able to find other nodes which are waiting to forward packets.

[MAL01] introduced several algorithms for applying the token ring to ad hoc networks such as the Local Recency (LR) algorithm. The LR algorithm will be used on a dedicated channel, i.e., a unique PN sequence is dedicated for the token ring.

A virtual ring is a ring that connects all the users together, satisfies the connectivity of the graph and minimizes the number of passages through each node. A round is completed when the entire group of users has been visited by the token. The following token ring algorithm implements a virtual ring.

3.4.2. Token Ring Algorithm

The leader node, i.e. the first receiver, initializes the token, communicating to its sender that the token is held. The other receivers sense the signal of the first sender and keep this path loss information. The leader node refers to its neighbors table, and sends the token to the other nodes following the LR-algorithm.

The other receiver nodes follow these instructions:

1. Waits for the reception of the token packet.
2. Communicates to its sender that the token is held.
3. The sender sends its identification to all receivers who, upon sensing it, obtain the path loss information.
4. The token holder puts its path loss information obtained previously onto the token.
5. Sends the token following the LR-algorithm.

After the initialization round, the token containing the path loss information is received by the leader node. By using the code initialization algorithm, the leader node initializes the combined PN sequences and transmission power assigned for each transmitter-receiver pair and puts this information onto the token. During the second round, each transmitter-receiver pair extracts their code assignment and transmission power. Subsequently, the token continually visits the receivers one by one, and in that way the path loss information is continuously shared amongst them. When a violation occurs, a

node, upon reception of the token, will correct its PN sequences by using the code correction algorithm and pass the token with its code assignment information.

3.5. Mitigating the MAI and the Fading Effects

In this section, we present a method to mitigate MAI and fading effects by using a code diversity technique.

A simple technique is deployed [SEK99]; the data information is sent through the different channels used by the transmitter-receiver pair. The received signals from the different branches are multiplied with the PN sequences allocated to them; each autocorrelation detector is synchronized with each transmitted signal. The output signal from the autocorrelation detector matched with the PN sequence $pn_{k,l}$ of the k^{th} transmitter is denoted by $Z_{k,k,pn_{k,l}}$ which is composed of the desired signal, MAI, noise, plus a fading component. The expression of the $Z_{k,k,pn_{k,l}}$ can be found in Section 2.1.1. In [SEK99], a weight $\alpha_{k,k,pn_{k,l}}$ is assigned to the l^{th} channel of the k^{th} transmitter; the channel with the greatest output is assigned the heaviest weight. $\alpha_{k,k,pn_{k,l}}$ are defined by

$$\alpha_{k,k,pn_{k,l}} = \frac{Z_{k,k,pn_{k,l}}{}^2}{\sum_{i=1..w_k} (Z_{k,k,pn_{k,l}})^2} \tag{3.14}$$

where w_k denotes the number of PN sequences assigned to the k^{th} transmitter-receiver pair. From (3.14), we can derive the weighted decision static $Z_{k,k}$ at the k^{th} receiver given by

$$Z_{k,k} = \sum_{l=1..w_l} \alpha_{k,k,pn_{k,l}} Z_{k,k,pn_{k,l}} \tag{3.15}$$

The k^{th} receiver uses a simple threshold decision given by the rule

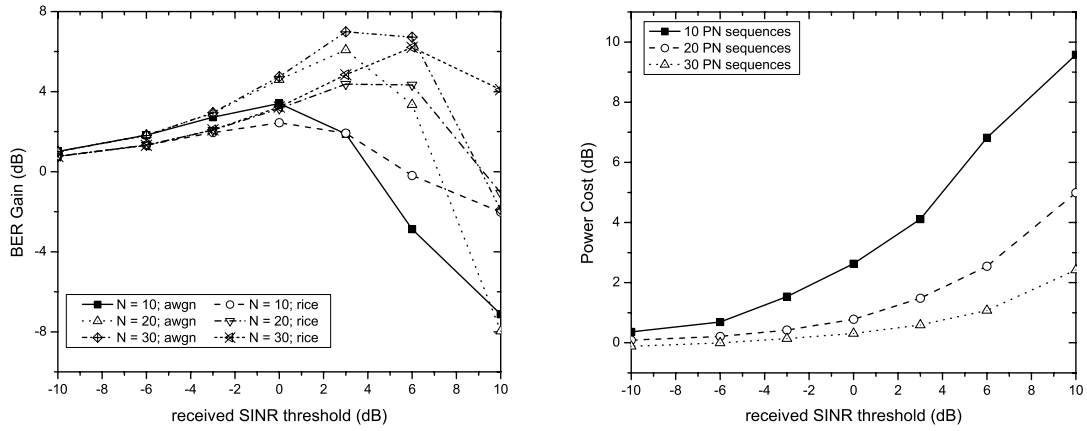
$$\widetilde{b_k(t)} = \begin{cases} -1 & \text{if } Z_{k,k} \le 0 \\ +1 & \text{if } Z_{k,k} > 0 \end{cases} \tag{3.16}$$

where $\widetilde{b_k(t)}$ is the data bit chosen after evaluation of $Z_{k,k}$. The case $Z_k = 0$ can be neglected due to its null probability.

3.6. Simulation and Performance

In our simulation, 12, 16, 24 and 32 pairs are drawn randomly on a rectangle of width and length of 100 units. The distance between each transmitter-receiver pair is chosen randomly to be less than 10 units. The reference gain G is chosen to be equal to 10 units^2. The G_{ij} are equal to the gain G divided by the square of the distance between the i^{th} transmitter and the j^{th} receiver. The variance of the white Gaussian Noise is set to 0.1, and the received SINR threshold of each receiver takes a value from the set $\gamma_k = \{-10, -6, -3, 0, +3, +6, +10\}$. The number of PN sequences assigned to each transmitter-receiver pair is 3. The PN sequences are the Walsh code sequences of length

64. The number of available sequences is chosen from the set $N = \{10, 20, 30\}$. To take into consideration, the effects of MAI in our simulations, a random delay is assigned to each interfering transmitter. Transmissions on an AWGN channel as well as a Rice channel of parameter $\sigma_a^2 = 0.1$ (average scattered power due to multipath) with BPSK modulation were simulated. The simulations were conducted using 1000 bits each trail, and each trial were repeated 100 times. We compare our multi-PN sequence scheme with the single branch model for which each transmitter-receiver pair is assigned a distinct PN sequence and uses power control for transmission.

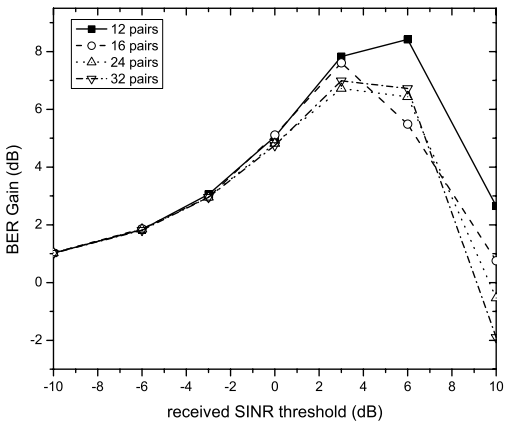


(a) **(b)**

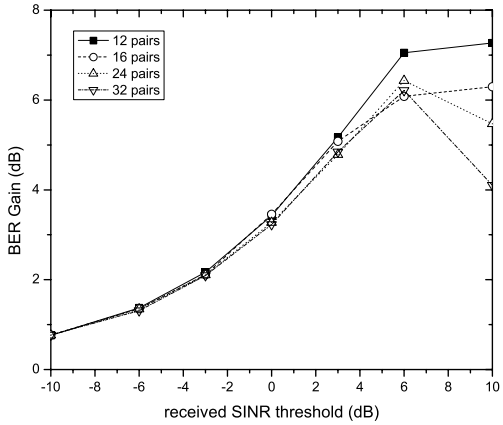
Figure 3-1 BER gain and power cost vs. received SINR threshold for 32 transmitter-receiver pairs

Figure 3-1 shows the BER gain and the transmission power cost by using our scheme over the single branch model for 32 transmitter-receiver pairs. In Figure 3.1 (a), the proposed code assignment reduces the BER in both conditions (AWGN and Rice) up to a certain point, $\gamma_k = 6\text{dB}$, beyond which, the BER increases, thus manifesting the so called near-far problem for CDMA. In the proposed scheme where the number of transmitters is

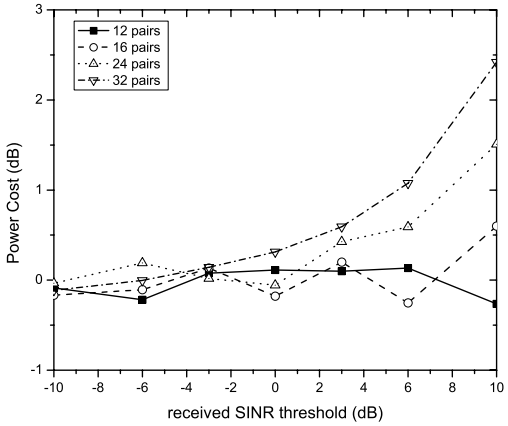
multiplied by 3, the near-far problem for CDMA is experienced for lower γ_k than the single branch system. In Figure 3-1 (b), this problem is emphasized. The total transmission power of the proposed system increases faster than the total transmission power of the single branch system. The increase in the number of PN sequences available in the system will reduce the number of collisions as this number increases with the number of transmitters in the network.



(a) AWGN channel



(b) Rice channel



(c)

Figure 3-2 BER gain and power cost vs. received SINR threshold for 30 PN sequences used

Figure 3-2 shows the BER gain and the power cost using 30 PN sequences for the code assignment. Through an AWGN and a Rice fading channel, the BER gain increases up to $\gamma_k = 6\text{dB}$, and beyond this point, this number decreases. As we can see, the proposed code assignment has a higher BER gain for a Rice fading channel where the code diversity scheme is used to mitigate fading effects. The power cost increases faster when the number of transmitter-receiver pairs increases, thus manifesting the near-far problem.

3.7. Conclusion

This chapter describes and analyzes a novel PN sequence assignment scheme for a DS-CDMA ad hoc network using code diversity technique. The advantage of this model compared to the scheme which assigns one PN sequence to each user, is the mitigation of the effect of MAI and fading and a flexible bit rate allocation. The proposed architecture ensures minimal total power consumption and a significant improvement of the BER for the entire system. The system is composed of two phases: the initialization and the correction phases.

We notice that our algorithm is more suitable for networks with slow or reasonable mobility like sensor networks and small sized networks to avoid excessive adjustments for the set of PN sequences and the power transmission of each transmitter-receiver pair. Unfortunately, because of its global approach, the complexity of our algorithm will increase with the number of users.

Chapter 4 Distributed Multi-Code Assignment for TOCA Network

In this chapter, we introduce an adaptive multi-code assignment scheme for DS-CDMA ad hoc networks. This scheme is using the proposed distributed Code Correction Protocol (CCP) to avoid the situation of having two nodes, which are two-hop apart, correcting their PN sequences at the same time. Subsequently, we introduce a new code assignment algorithm which assigns for each node the "best PN sequence" to limit collisions and MAI.

Section 4.1 introduces the concept of multi-code assignment for a TOCA system. Section 4.2 presents initialization protocols for a TOCA system. Section 4.3 describes existing protocols used during the code correction process in a TOCA system. Section 4.4 presents the distributed CCP protocol for the correction process. Section 4.5 discusses the complexity of the proposed protocol. Section 4.6 presents a new code assignment algorithm which limits the number of packets loss due to collisions and MAI. In Section 4.7, the simulation for the ad hoc network using the CCP protocol is presented. Section 4.8 and Section 4.9 discusses respectively the impact of the transmission power and the velocity on the performance of the system. Finally, Section 4.10 concludes this chapter.

4.1. Multi-Code Assignment for a TOCA System

In Section 2.2.1, we showed that TOCA and POCA schemes are the only two code assignment schemes able to eliminate the hidden and exposed terminal problems. We have also demonstrated that the TOCA scheme has a simpler implementation than the POCA scheme. For these reasons, the TOCA scheme has attracted our attention for distributed ad hoc networks.

4.1.1. The TOCA Layering Model

In the simplified OSI model, a mobile host can be represented by 4 layers: the Physical, the MAC, the Routing, and the Application layer. In a TOCA network, the color module is implemented at the MAC level and is used to ensure that the node has no violation with its 2-hop neighbors. Figure 4-1 depicts the modified OSI model for a TOCA scheme.

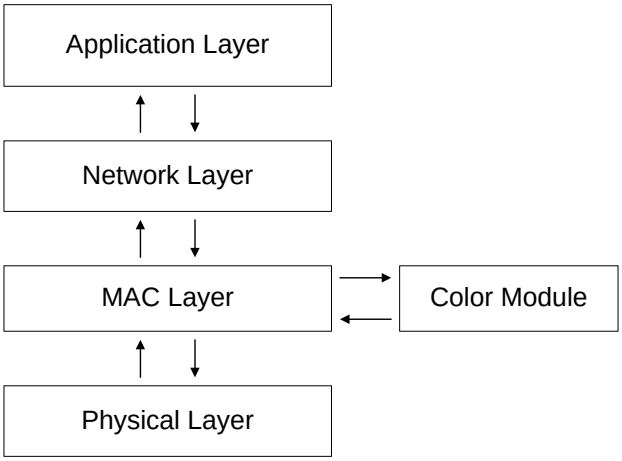


Figure 4-1 - Simplified OSI Model for a TOCA scheme

4.1.2. Number of PN Sequences Assigned per Transmitter

For a non-adaptive code assignment scheme, the number of PN sequences assigned for each node is decided in advance and does not vary with the condition of the network.

In an adaptive code assignment scheme, the number of PN sequences to be assigned for each transmitter depends on its needs. These needs represent the quality of service which must be provided for each user by the adaptive TOCA scheme. They can be quantified as the number of packets loss, the packet transmission delay, the number of neighbors, or the bandwidth needed.

4.1.3. Information Storage for TOCA

In a TOCA scheme, each node needs to know the code assignment information concerning itself, its one hop and its two-hop neighbors. Each 1-hop table element contains the node's address, priority, PN sequences used for transmission, its expiration, and the node's 1-hop neighbors information (address, priority and PN sequences). Figure 4-2 presents the data storage information for a TOCA scheme.

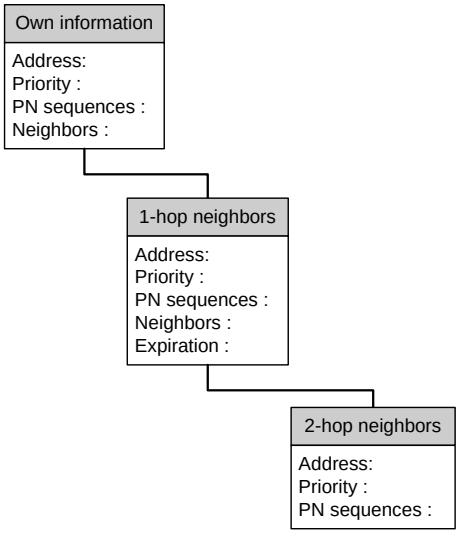


Figure 4-2 - TOCA data storage

The node stores the information concerning itself, its 1-hop and its 2-hop neighbors. Therefore, the maximal number m_e of table entries which must be stored by each node for a TOCA scheme is given by

$$\begin{aligned} m_e &= \Delta(\Delta-1)+\Delta+1 \\ &= \Delta^2+1 \end{aligned} \tag{4.1}$$

where Δ is the maximal degree of the ad hoc network.

4.2. Code Initialization Protocol

Code initialization has been widely studied in [HU93, MAK87, KIM91, HUN92, BER95 CID89, and GAR97]. It is employed to initialize the TOCA ad hoc network by assigning each user a PN sequence for transmission. Upon its completion, communications between

nodes can start. Thus, this process must be quick and have low complexity due to its one time usage.

4.2.1. Random Initialization

In the random initialization process, each node selects randomly its PN sequences from the set of code sequences available $C = \{c_j\}_{j=1}^{N}$ and subsequently informs its 1-hop and 2-hop neighbors of its choice. The random initialization is a fast and simple process but only gives a rough code assignment where many violations may occur. When the random code initialization is completed, violating nodes move to the correction process to re-adjust their PN sequences for transmission. Due to its zero-complexity, the random initialization has been chosen for our protocol.

4.2.2. Least PN Sequences Algorithms

In [MAK87, KIM91, BER95], many algorithms were presented for the code initialization process. The objective of those algorithms is to find a code assignment using the least number of PN sequences N which satisfies the condition in Section 2.2.2.3. The code initialization process completely eliminates the hidden terminal problem but has the disadvantage of being time and memory consuming. Some of those algorithms are described in Section 2.3.

4.2.3. Sink-Tree Coloring Algorithm

[HU93] introduced the sink-tree coloring algorithm as a code initialization process. The distributed algorithm gives a quick but rough code assignment for the network which can reduce the possible number of violations compared to the random initialization process. The algorithm supposes the existence of a *k*-hop leader election [NAK00, NAK02] where *k* denotes the maximal number of hops between any node and its leader. First, the leaders randomly initialize their PN sequences. The leaders then propose to each of their 1-hop neighbors a set of PN sequences which limit the number of violations. The initialization (INIT) packets, which contain the code assignment information of its originator as well as the proposed code assignment for its 1-hop neighbors, are broadcasted to their neighbors. The initialization chain thus starts.

When an unassigned node receives an INIT packet, it selects the proposed code sequences from the INIT packet. The node then broadcasts the INIT packet containing the code assignment information for its unassigned 1-hop neighbors. The initialization process is completed when every node is assigned a code sequence. The sink tree algorithm is executed in a parallel and distributed manner with the advantage of not being time consuming. However, this process presents two drawbacks. First, the algorithm is a relatively complex mechanism involving a specific control packet for a one-time usage. Second, a *k*-hop leader election is needed to ensure that the entire network will be initialized. The sink tree algorithm is presented in Figure 4-3.

```
void Color::CodeInitialization()
    for (each table Element of the 1-hop table)
        TableElement* e = new TableElement();
          if (e has no color)
                  if (e is a 2 hop neighbors)
                          choose randomly a color which is not used by a 1-hop neighbors or itself  from the set C;
                  else
                          choose randomly a color which is not used by a 1-hop neighbors from the set C ;
```

Figure 4-3 - Sink tree algorithm

Where the number of PN sequences available *N* is assumed to satisfy the condition given by the theorem 2 in Section 2.2.2.3.

4.3. Existing Code Assignment Protocols

The code assignment protocol controls and corrects PN sequence violations in the ad hoc network. For an effective code sequence correction, it is primordial to avoid the situation of having two nodes, which are two-hop apart, changing their PN sequences at the same time which a deadlock may occur.

4.3.1. Highest Priority Approach

A simple code assignment protocol has been introduced by Garcia-Luna-Aceves [GAR97]. It requires the perfect capture of code assignment tables and that *N* is at least $\Delta(\Delta-1)+1$. When a pair of 2-hop neighbors is sharing the same PN sequence, the highest priority node will re-adjust its PN sequence while the other violating node keeps it. The main drawback is that a pair of violating 2-hop neighbors may not share the same

PN sequence and in this case if the pair is choosing simultaneously the same sequence, another violation will occur.

4.3.2. Chain Re-Coloring Approach

Hu [HU93] introduced a chain re-coloring approach to prevent a pair of 2-hop neighbors from correcting their PN sequences simultaneously. A conflicting node will collect chain information concerning its violating 2-hop neighbors and will wait for the violating 2-hop neighbors with higher priority to correct their PN sequence before starting its own correction. However, this approach has two drawbacks. The first drawback is the collection of correction chain amongst violating nodes. The second drawback is that during the correction process the chain information earlier collected might be inaccurate. This might be experienced in the case of long correction chain in a frequent changing topology.

4.4. The Code Correction Protocol (CCP)

The different drawbacks from the two existing protocols for TOCA mobile ad hoc networks have motivated our work to develop the Code Correction Protocol (CCP) for the correction process. The CCP protocol is a competitive-based protocol more dynamic than the chain re-coloring protocol which ensures that a pair of violating 2-hop neighbors is not correcting its PN sequences simultaneously. The CCP protocol combined with the adaptive multi-code assignment algorithm presented in Section 4.6 introduces the first solution to the adaptive multi-code assignment problem.

4.4.1. CCP Description

We first define the two time constants $t0$ and $t1$, respectively representing the maximum transmission delay and the maximum random time between transmissions of two distinct control packets. $M_x(A)$ is the ensemble of x-hop neighbors of node A previously defined in Section 2.2.2.1. We assume that control packets are transmitted over a CCA channel using a smaller bandwidth independent of the data channel in order to keep nodes reactive enough to topology changes. Therefore, the node can for example transmit a data packet and at the same time receive or transmit a control packet. The narrowband control channel and the wideband data channel are not overlapping in frequency to avoid interference between them. A random access ALOHA with the parameter $t1$ is used to limit control packet collisions.

The CCP protocol is composed of two phases.

The 1st phase consists for nodes to keep track of changes of the code assignment of their 1-hop and 2-hop neighbors. *The Code Assignment Message* (CAM) has been modified from [GAR97] for this purpose. The CAM packet contains the source information similar to [GAR97] and its 1-hop neighbors' information. The CAM packet is sent periodically like a conventional HELLO packet. When a violation is sensed, the node moves to the 2nd phase.

In the 2nd phase which corresponds to the correction phase, a violating node, say node A, must ensure that none of its 2-hop neighbors is correcting its PN sequences simultaneously. Node A initializes a correction by broadcasting a *Pre-Code Assignment Message* (PCAM), and subsequently, waits for the collection of the *Pre-Code Assignment Message Acknowledgments* (PCAM_ACK) from $M_1(A)$ during the *Wait_For_PCAM_ACK* period. The node B of $M_1(A)$, which has received the PCAM packet from node A, will generate a PCAM_ACK packet to inform $M_1(B)$ about the possible agreement of node B with the correction of node A. Node A is said to be the *locker* of node B, if node B has agreed with the correction of node A for the *Locked_Time* period. During this period, node B can not be locked by another node. Now supposing that node B has received PCAM packets from several nodes of $M_1(B)$ during the *Locked_Time* period, the node with the highest priority which has transmitted a PCAM packet to node B is called the *HighestPriority* node of node B. The PCAM packet processing is depicted in Figure 4-5. When node A has locked $M_1(A)$ it can proceed to the correction of its PN sequences and broadcast a CAM packet to $M_1(A)$ and $M_2(A)$. Table 4-1 presents the structure of the CCP messages and Figure 4-4 illustrates the CCP mechanism.

Table 4-1-CCP messages structure

CAM	PCAM	PCAM_ACK
scrAddress scrPriority scrPNSequences Neighbors • Address • Priority • PNSequences	scrAddress scrPriority	scrAddress lockerAddress HighestPriorityAddress HighestPriority

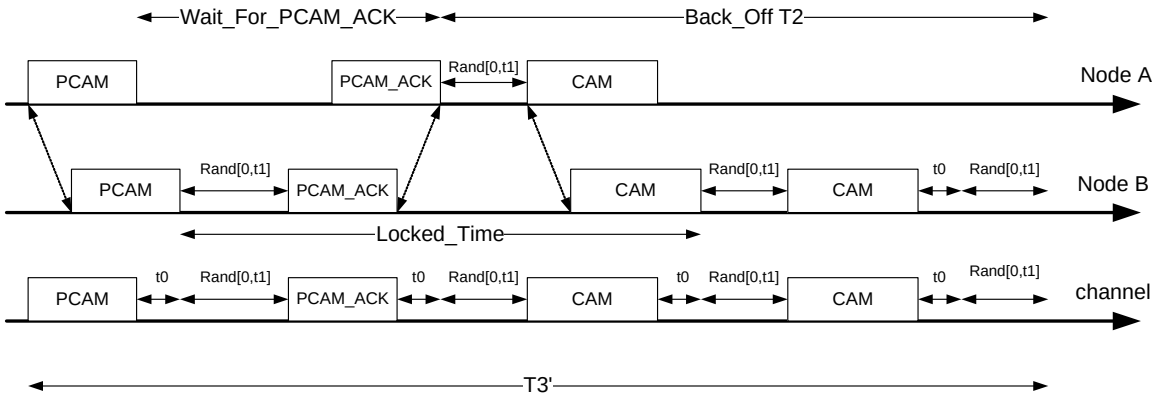


Figure 4-4 – CCP mechanism

1. When a PCAM packet is received
   → If the node has not been locked, initialize the parameters:
   - locked for the Period *Locked _ Time*
   - locked=true;
   - locker=scrAddr;
   - HighestPriority=scrPriority;
   - HighestPriorityAddr=scrAddr;

   → Else,
   - locked=true;
   - locker=scrAddr;
   - if(scrPriority>HighestPriority)
     - HighestPriority=scrPriority;
     - HighestPriorityAddr=scrAddr;
   - elseif(scrPriority=HighestPriority ∩ HighestPriorityAddr<scrAddr)
     HighestPriorityAddr=scrAddr;
   - else
     keep the same PCAM_ACK parameters;

2. Generate and Broadcast the PCAM_ACK packet.

Figure 4-5- PCAM packet processing

In CCP, each node follows the 5 back off rules to ensure efficient correction amongst violating nodes. This procedure guarantees that at least the violating nodes corresponding to the *HighestPriority* nodes for all their 1-hop neighbors correct their PN sequences.

These rules are given below according to the 4 back off time constants T1, T2, T3 and T4.

1st rule: Upon sensing a violation and if no code sequence is available for the correction, the node backs off for the period *T4.* If a PCAM_ACK packet is received during this period and the remaining back off time is shorter than the period *T1,* the node will back off for another period *T1*.

2nd rule: When no correction has been attempted and a PCAM_ACK packet is received, the node backs off for the period *T1* where the nodes will not attempt any future correction. If another PCAM_ACK packet is received during this period, the node will back off for another period *T1*.

3rd rule: At the end of the *Wait_For_PCAM_ACK* period, a node which attempted and failed to lock all its 1-hop neighbors will back off for the period *T2* if it is the *HighestPriority* node of all its 1-hop neighbors.

4th rule: At the end of the *Wait_For_PCAM_ACK* period, a node which attempted and failed to lock all its 1-hop neighbors, will back off for the period *T3* if it is not the *HighestPriority* node of all its 1-hop neighbors.

5th rule: At the end of a back off period and if a "better" code assignment exists, the node attempts a new correction by sending a PCAM packet. The concept of "better" code assignment will be introduced in Section 4.6.

With reference to Figure 4-4 and the 5 back off rules, the abovementioned time constants are given by

$$Wait\_For\_PCAM\_ACK = 2 \times t0 + t1 + \partial_{PCAM\_ACK} \tag{4.2}$$

$$Locked\_Time = 2 \times t0 + 2 \times t1 + \partial_{PCAM\_ACK} + \partial_{CAM} \tag{4.3}$$

$$T2 = 3 \times t1 + 2 \times t0 + 2 \times \partial_{CAM} \tag{4.4}$$

$$T3' = 4 \times t0 + 4 \times t1 + \partial_{PCAM} + 2\partial_{CAM} + \partial_{PCAM\_ACK} \tag{4.5}$$

$$T3 = T2 + T3' \tag{4.6}$$

$$T1 = 2 \times T3' - \partial_{PCAM} - \partial_{PCAM\_ACK} \tag{4.7}$$

where ∂_X is defined as the time duration (ratio between the size of the packet and bit rate of the channel) of the packet X. The period *T3'* represents the period where a node corrects its violation after having successfully locked its 1-hop neighbors. The back off

period *T4* is chosen independently from the other time constants. The entire CCP protocol is described in Figure 4-6 and Figure 4-7.

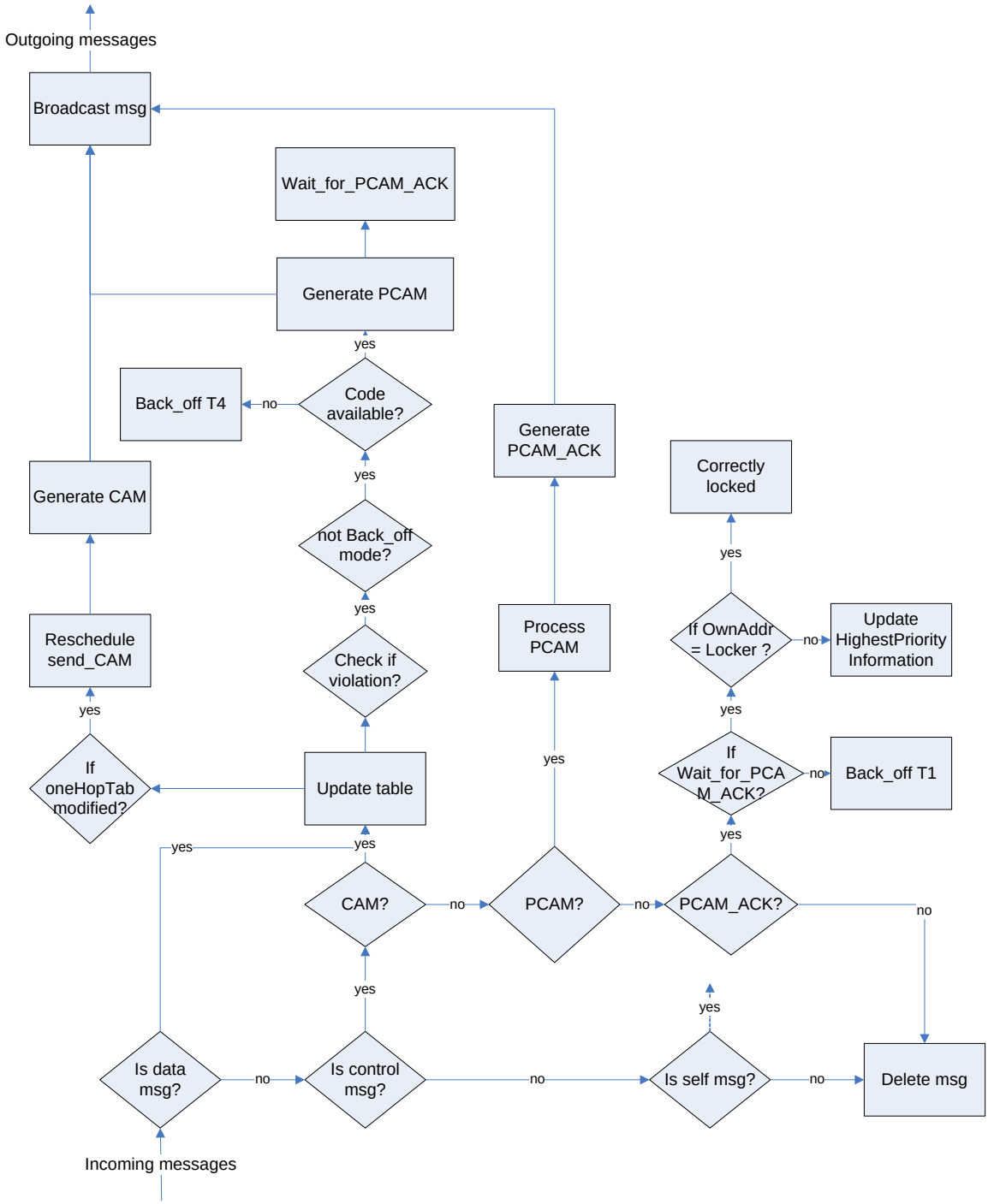


Figure 4-6 - Control and data messages management

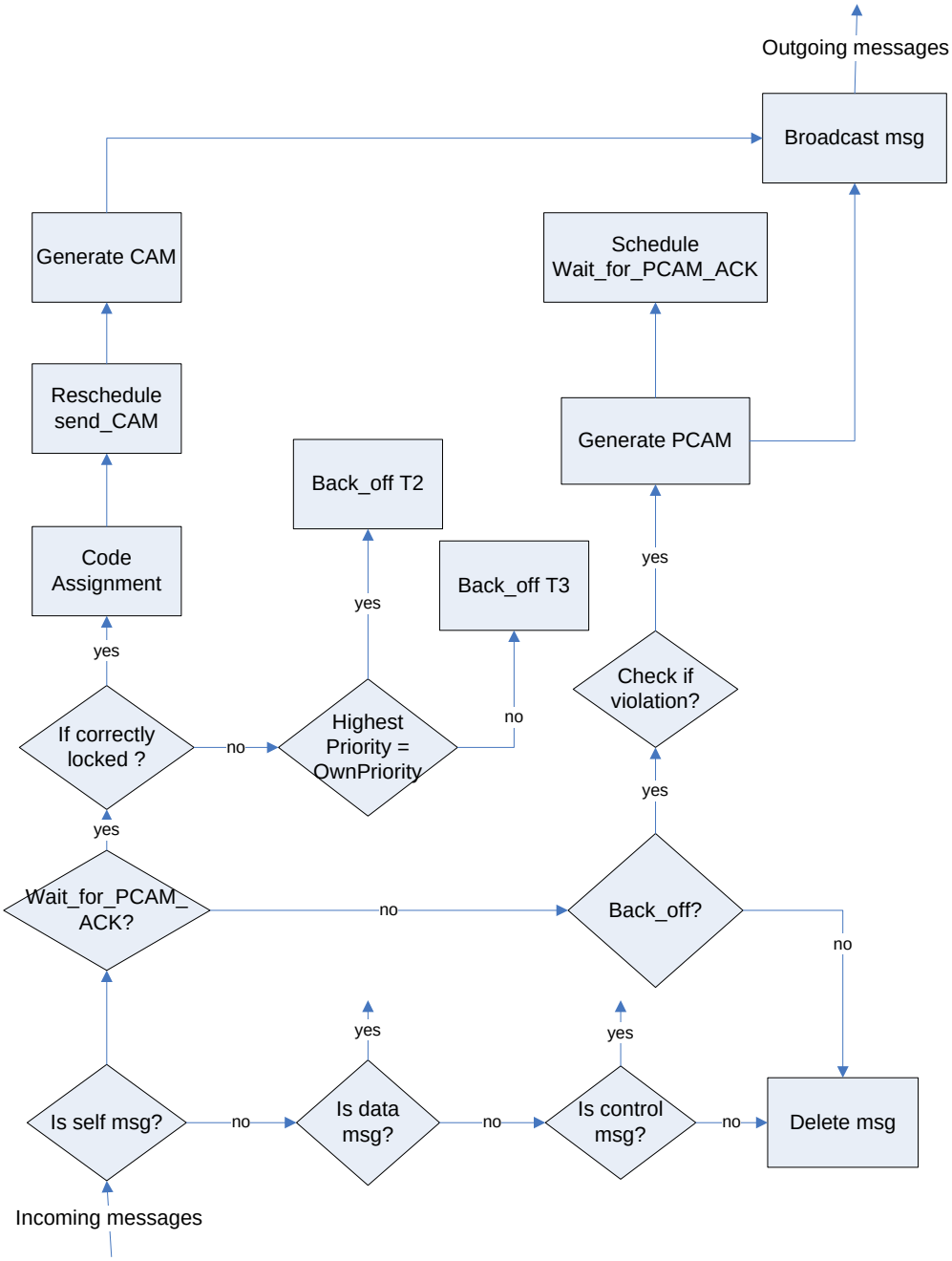


Figure 4-7 - Self messages management

4.4.2. Correctness of the Protocol

To prove the correctness of the protocol, a few assumptions are made:

i. When two nodes A and B have the same priority, node A is said to have a higher priority than B, if the A's ID is lower B's ID.

ii. A node which is correcting its PN sequences has an accurate coloring table.

iii. During the time $2 \times T3'$ when a node is correcting its PN sequences; there is no change of connectivity with its 1-hop and 2-hop neighbors.

Theorem 1: *If a pair of violating 2-hop neighbors is correcting their violations sequentially, then no deadlock will occur.*

Proof: Suppose that a node is correcting its PN sequences during *T3'*, after this period all its 1-hop and 2-hop neighbors will be informed of its new code assignment. Subsequently, all its 2-hop neighbors can proceed to a correction without a deadlock. The theorem 1 follows.

Theorem 2*: The highest priority node of a correction chain is the HighestPriority node of all its 1-hop neighbors.*

Proof: A correction chain is defined by violating nodes separated in pairs by 2-hops and have attempted a correction by sending a PCAM packet. Suppose that the highest priority node of the chain which is also by definition the highest priority node amongst its violating 2-hop neighbors including itself has sent a PCAM packet, then given the 3rd back up rule, the Theorem 2 follows.

The converse is not true in general. This can be shown with an example. Supposing a chain with 4 elements with priority 2-1-2.5-2, the element on the right is the *HighestPriority* node of all its 1-hop neighbors, however is it is not the highest priority of the chain.

Theorem 3*: The HighestPriority node of all its 1-hop neighbors is guaranteed to successfully correct its PN sequences after the period* $2 \times T3'$.

Proof: If the *HighestPriority* node has transmitted a PCAM packet, two cases can happen after the *WAIT_FOR_PCAM_ACK* period: 1) the node has locked all its 1-hop neighbors, or 2) the node has not locked all its 1-hop neighbors. In the first case, the node proceeds to transmit a CAM packet to its 1-hop and 2-hop neighbors during the period *T2*. In the second case, given the 2^{nd}, 3^{rd} and 4^{th} back up rules, the *HighestPriority* node backs off for the period *T2* while its violating 2-hop neighbors which have failed to locked their 1-hop neighbors for the period *T3* and the other 2-hop neighbors for the period *T1*. Thus, the *HighestPriority* node is the only node to transmit a PCAM packet amongst its 2-hop neighbors including itself and correct its PN sequence during the period *T3'* and the Theorem 3 follows.

As we can see, CCP is a competitive-based protocol which does not require any prior information for the correction process. When violations occur, nodes compete to lock their 1-hop neighbors; the nodes which succeed can subsequently correct their PN sequences and the nodes which fail back off to give priority to the highest priority nodes

to correct their PN sequences. Corrections are done sequentially until violations are completely eliminated. The back off strategy used improves the code assignment by allowing the nodes which have a code sequence available for correction to correct their violation. In a frequent topology changing ad hoc network, CCP is more efficient than the chain re-coloring approach.

4.4.3. Example

In this section, CCP is presented in the example given by Figure 4-8. We consider a network constituting of 10 nodes in which 4 nodes are under violation. The node with lower ID had a higher priority than a node with greater ID.

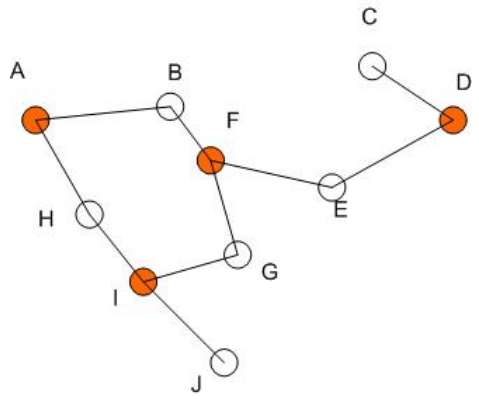


Figure 4-8 - Example of the correction process using CCP

1st Attempt at t=0, A, F, I and D initialize their PCAM packet and after the *WAIT_FOR_PCAM_ACK* period, the PCAM originators receive PCAM_ACK packets from their 1-hop neighbors with the following information

Table 4-2- PCAM_ACK information after WAIT_FOR_PCAM_ACK at 1st attempt

PCAM Originator	PCAM_ACK Originator (From)	Locker	Highest Priority	Highest Priority Address
A	B(A), H(A), B(F), H(F)	A, A, A, A	A, A, A, A	A, A, A, A
F	B(A), B(F), E(F), G(F), G(I), E(D)	A, A, F, F, F, F	A, A, F, F, F, D	A, A, F, F, F, D
I	H(A), G(F), J(I), H(I), G(I)	A, F, I, A, F	A, F, I, A, F	A, F, I, A, F
D	E(F), C(D), E(D)	F, D, F	F, D, D	F, D, D

A is the only node to successfully lock its 1-hop neighbors; it proceeds to the correction of its PN sequences and transmits a CAM packet. D is the *HighestPriority* node for $M_1(D)$ backs off for the period *T2*. F and I which are not the *HighestPriority* nodes of all their 1-hop neighbors back off for the period *T3*.

At $t = T3'$, D transmits a PCAM packet and receives PCAM_ACK packets with the information

Table 4-3-PCAM_ACK information at *t = T3'*

PCAM Originator	PCAM_ACK Originator (From)	Locker	Highest Priority	Highest Priority Address
D	E(D), C(D)	D, D	D, D	D, D

D successfully locks its 1-hop neighbors, corrects its PN sequences and transmits a CAM packet.

2nd Attempt at $t = 2 \times t3'$, F and then I transmit their PCAM packet and receive PCAM_ACK packets with the information

Table 4-4- PCAM_ACK information at the 2nd attempt

PCAM Originator	PCAM_ACK Originator (From)	Locker	Highest Priority	Highest Priority Address
F	B(F), E(F), G(F), G(I)	F, F, F, F	F, F, F, F	F, F, F, F
I	G(F), J(I), H(I), G(I)	F, I, I, F	F, I, I, F	F, I, I, F

F successfully locks its 1-hop neighbors, subsequently proceeds to the correction of its PN sequences, and finally, generates its CAM packet. The correction of F will correct at the same time the code violation of I. However, I still backs off for another *T3.* The correction of the violating nodes is fully completed at $t = 3 \times T3'$.

4.5. Complexity of a Correction Chain of Length L

To evaluate the complexity of the proposed algorithm, we want to estimate the number of control messages exchanged and the correction time needed to correct a correction chain of length L. The worst case will give an upper bound of the two numbers. Consider a chain with nodes sorted in increasing priorities, in which the highest priority node amongst its violating 2-hop neighbors including itself is the highest priority element of the correction chain, and suppose that during their first attempt to lock their one-hop neighbors, none of them succeeds. Consequently, there is only one correction after the period $2 \times T3'$. At the first attempt, *L* PCAM packets are sent and each node which receives the PCAM will generate a *PCAM_ACK* packet. At the second attempt, only the

highest priority node locks the channel by sending a PCAM packet and subsequently broadcasts a CAM packet to its 1-hop and 2-hop neighbors. Therefore, the number n_L of control messages exchanged in the network for the correction of one element of the chain if $L>1$ is given by

$$\begin{aligned} n_L &= L+\Delta.L+1+\Delta+1+\Delta \\ &= (\Delta+1).(L+2) \end{aligned} \tag{4.8}$$

where Δ is the degree of the network as previously defined.

The total number of control messages N_L sent for a chain of length $L>1$ is given by

$$\begin{aligned} N_L &= \sum_{l=2..L} n_l \\ &= \frac{(\Delta+1)(L+4)(L-1)}{2} \end{aligned} \tag{4.9}$$

The total number of control messages N_L exchanged among nodes for the correction of a chain of length *L* is thus upper bounded by $\mathrm{O}\left(\frac{(\Delta+1)}{2}.L^2\right)$.

The maximum time T_L required for the correction process to be completed, given a chain of length *L* is

$$T_L = 2.(L-1).T3' \tag{4.10}$$

4.6. TOCA and Code Assignment Algorithm

In the previous section, we have presented the CCP protocol that ensures that only one node is changing its set of PN sequences amongst its two-hop neighbors including itself. This function is vital for an effective code assignment. Hu [HU93] and Low [LOW03] have seen that a perfect dimensioning of the ensemble *C* of PN sequences available is difficult to obtain in a mobile ad hoc network where the degree of the network Δ varies frequently. The dimensioning becomes yet impossible for an adaptive code assignment in which Quality of Service (QOS) is considered. In addition, the set *C* is usually fixed in advance and limited by the specifications set by the manufacturer of the mobile device and not by the administrator of the ad hoc network if it exists.

4.6.1. Definitions, Assumptions and Goals

We consider *K* identical nodes in the network. Each node has knowledge of the set $C = \{c_i\}_{i=1}^{N}$ of PN sequences available as well as the average cross-correlation matrix $CC(N \times N)$. Each element of the matrix $\left(cc_{i,j}\right)_{N\times N}$ is defined as follows

$$\begin{cases} cc_{i,j} = \dfrac{1}{6M^3} r_{c_i,c_j} & \forall (i,j) \in N \times N \\ cc_{i,j} = cc_{j,i} \end{cases} \tag{4.11}$$

where M is the length of the PN sequences and r_{c_i,c_j} represents the average cross correlation function between the sequence c_i and c_j.

If the nodes are using the CCP protocol, we have shown that at most one user is correcting its PN sequences amongst its 2-hop neighbors including itself. We assume that received signals from the transmitters at a distance of 2-hop and beyond are negligible compared to received signals from transmitters 1-hop away. We assume also that during the correction process, the node has perfect knowledge of the code assignment of its 1-hop and 2-hop neighbors.

The goal of the code assignment algorithm is to assign to each user the "best PN sequence possible" in order to limit collisions as well as MAI.

4.6.2. The Collision Cost Function

The correcting node will choose the "best PN sequences" in order to limit the number collisions. As we can see, for a given node, the code assignment of each of its two-hop neighbors has a direct effect on its number of collisions. Consider the worst case scenario where node *j* is transmitting to node *l* and all the 1-hop neighbors of *l* are simultaneously transmitting, as depicted by Figure 4-9.

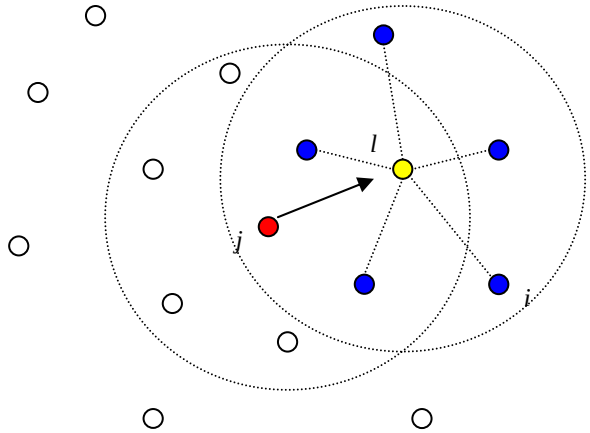


Figure 4-9 Maximum collisions and interferences at reception

We give a simple heuristic approach to solve the code correction problem. As we have seen in Section 2.2.1.4, the receiver is made of a bank of matched filters, each one being matched to a particular PN sequence. A collision occurs, when a node has been synchronized to one packet on a given channel and at the same time another packet is received through the same channel. In a synchronous system, both packets will collide causing both to be dropped. In an asynchronous system, only the last incoming will be dropped. In either case, node *j* which is changing its PN sequences should be the PN sequences least used by the neighbors of node *l* to limit collisions at that node. To express this condition mathematically, we use the code assignment vector $\Psi_k = \{\varphi_{ki}\}_{i=1}^{N}$ associated to the k^{th} transmitter given by

$$\varphi_{k,i} = \begin{cases} 1 & \text{if } c_i \in PN_k \\ 0 & \text{otherwise} \end{cases} \tag{4.12}$$

We can define the collision cost vector $\Pi_{j,l}$ associated to the j^{th} transmitter and the l^{th} receiver as the sum of the code assignment vector for each k^{th} transmitter $(k \neq j)$ neighbor of the l^{th} receiver, i.e.

$$\Pi_{j,l} = \sum_{\substack{k \in M_1(l) \\ k \neq j}} \Psi_k \tag{4.13}$$

where $M_x(l)$ is the ensemble of x-hop neighbors of the l^{th} node as previously defined. The overall collision cost vector associated to the j^{th} transmitter is given by

$$\begin{aligned} \Pi_j &= \sum_{l \in M_1(j)} \Pi_{j,l} \\ &= \sum_{l \in M_1(j)} \sum_{\substack{k \in M_1(l) \\ k \neq j}} \Psi_k \end{aligned} \tag{4.14}$$

Finding the best PN sequences possible to minimize the number of collisions is then reduced to select the PN sequences corresponding to the lowest collision cost elements. A simple heuristic is thus obtained

1. Compute the collision cost vector Π_j
2. Select randomly the PN sequences corresponding to the lowest elements of Π_j.
3. Update the set $PN_j = \left\{ pn_{j,n} \right\}_{n=1}^{w_j}$ of PN sequences assigned to the j^{th} transmitter.

4.6.3. The MAI Cost Function

In the previous section, we introduced a new heuristic which finds the best set of PN sequences to minimize collisions. The code assignment algorithm can be improved in order to minimize the undesirable effect of MAI when possible. Choosing a PN sequence for a transmitter has its influence on the MAI level at each of its 1-hop neighbor receivers.

From Figure 4-9, suppose that node *j* is changing its code sequence $pn_{j,1}$, we can express the worst case SINR at the l^{th} receiver by choosing the PN sequence $pn_{j,1}$ for the j^{th} transmitter as

$$SINR_{l,j,pn_{j,1}} = \frac{P_{j,l}}{\frac{1}{6M^3}\cdot \sum_{k\in M_1(l)} \sum_{\substack{1\le m\le w_k \text{ if } k\neq j \\ 2\le m\le w_k \text{ if } k=j}} P_{k,l} r_{pn_{k,m},pn_{j,1}} + \frac{N_0}{T_b}} \tag{4.15}$$

where $P_{j,l}$, N_0, T_b, M and $r_{pn_{k,m},pn_{j,1}}$ are as defined in Section 2.1.2 and w_k is the number of PN sequences assigned to the k^{th} transmitter.

Thus, the additional MAI at the l^{th} receiver due to choosing $pn_{j,1}$ for the j^{th} transmitter in the $SINR_{l,j,pn_{j,1}}$ is given by

$$MAI_{l,j,pn_{j,1}} = \frac{1}{6M^3} \cdot \sum_{k \in M_1(l)} \sum_{\substack{1 \le m \le w_k \ \text{if} \ k \ne j \\ 2 \le m \le w_k \ \text{if} \ k = j}} P_{k,l} r_{pn_{k,m}, pn_{j,1}} \tag{4.16}$$

Now if we suppose that the l^{th} receiver is matched to the PN sequence $pn_{k,m}$ of the k^{th} transmitter, by deriving the expression of $MAI_{l,k,pn_{k,m}}$, we can derive the additional interference created by the j^{th} transmitter by choosing $pn_{j,1}$ is given by

$$\Phi_{l,k,pn_{k,m},j,pn_{j,1}} = \frac{1}{6M^3} . P_{j,l} r_{pn_{j,1}, pn_{k,m}} \tag{4.17}$$

The overall additional interference at the l^{th} receiver $\mathrm{T}_{j,l,pn_{j,1}}$ due to choosing $pn_{j,1}$ for the j^{th} transmitter is then given by

$$\begin{aligned} \mathrm{T}_{j,l,pn_{j,1}} &= MAI_{l,j,pn_{j,1}} + \sum_{k \in M_1(l)} \sum_{\substack{1 \le m \le w_k \ \text{if} \ k \ne j \\ 2 \le m \le w_k \ \text{if} \ k = j}} \Phi_{l,k,pn_{k,m},j,pn_{j,1}} \\ &= \frac{1}{6M^3} \cdot \sum_{k \in M_1(l)} \sum_{\substack{1 \le m \le w_k \ \text{if} \ k \ne j \\ 2 \le m \le w_k \ \text{if} \ k = j}} P_{k,l} r_{pn_{k,m}, pn_{j,1}} + \frac{1}{6M^3} \cdot \sum_{k \in M_1(l)} \sum_{\substack{1 \le m \le w_k \ \text{if} \ k \ne j \\ 2 \le m \le w_k \ \text{if} \ k = j}} P_{j,l} r_{pn_{j,1}, pn_{k,m}} \\ &= \frac{1}{6M^3} \sum_{k \in M_1(l)} \sum_{\substack{1 \le m \le w_k \ \text{if} \ k \ne j \\ 2 \le m \le w_k \ \text{if} \ k = j}} \left(r_{pn_{k,m}, pn_{j,1}} \left(P_{k,l} + P_{j,l} \right) \right) \end{aligned} \tag{4.18}$$

By summing over all the $l \in M_1(j)$, we can extract the overall additional interference at $\mathrm{T}_{j,pn_{j,1}}$ associated to j^{th} the transmitter by choosing $pn_{j,1}$, given by

$$\mathrm{T}_{j,pn_{j,1}} = \frac{1}{6M^3} \sum_{l \in M_1(j)} \sum_{k \in M_1(l)} \sum_{\substack{1 \le m \le w_k \text{ if } k \ne j \\ 2 \le m \le w_k \text{ if } k = j}} \left(r_{pn_{k,m}, pn_{j,1}} \left(P_{k,l} + P_{j,l} \right) \right) \tag{4.19}$$

We denote by $\mathrm{T}_j = \left\{ \mathrm{T}_{j,c_i} \right\}_{i=1}^{N}$ the overall additional interference vector associated to j^{th} transmitter given by

$$\mathrm{T}_j = \sum_{l \in M_1(j)} \sum_{k \in M_1(l)} \left(\left(P_{k,l} + P_{j,l} \right) CC.\Psi_k \right) \tag{4.20}$$

where the $N \times N$ matrix CC and the vector Ψ_k are defined respectively in (4.11) and section 4.6.2.

Making the reasonable assumption that the received power level for each transmitter-receiver pair is identically and randomly distributed with mean $\mathrm{E}\left(P_{j,l} \right) = P_{mean}$ and that the $P_{j,l}$ are bounded by two positive constants (the case in which the distance between the transmitter-receiver null is excluded). We can derive $\mathrm{E}\left(\mathrm{T}_j \mid \left\{ P_{j,l} \right\} \right)$ as the mean vector T_j conditioned by the received power levels $P_{j,l}$ by

$$\mathrm{E}\left(\mathrm{T}_j \mid \left\{ P_{j,l} \right\} \right) = \sum_{l \in M_1(j)} \sum_{k \in M_1(l)} \left(\left(P_{k,l} + P_{j,l} \right) CC.\Psi_k \right) \tag{4.21}$$

By taking the expectation over the received power levels $P_{j,l}$, we can expressed the mean value $\mathrm{E}(\mathrm{T}_j)$ as

$$\begin{aligned}\mathrm{E}(\mathrm{T}_j) &= \sum_{l\in M_1(j)}\sum_{k\in M_1(l)}\left(\left(\mathrm{E}(P_{k,l})+\mathrm{E}(P_{j,l})\right)CC.\Psi_k\right)\\ &= 2.P_{mean}.CC.\left(\sum_{l\in M_1(j)}\sum_{k\in M_1(l)}[\Psi_k]\right)\\ &= 2.P_{mean}.CC.(\Pi_j+\Psi_j)\\ &= 2.P_{mean}.\overline{\mathrm{T}}_j\end{aligned} \tag{4.22}$$

where $\overline{\mathrm{T}_j}$ represents the simplified MAI cost vector associated to the j^{th} transmitter. The PN sequence corresponding to the lowest element $\overline{\mathrm{T}_j}$ is chosen for the code assignment. This code sequence minimizes the MAI associated to the j^{th} transmitter.

To take into consideration, the effects of the MAI during the code assignment, the following heuristic algorithm may be used

1. Compute the Π_j and $\overline{\mathrm{T}_j}$ cost vectors.
2. Select the PN sequence $c_x \notin PN_j$ corresponding to the lowest elements of Π_j. If more than one PN sequence exist, select the PN sequence corresponding to the lowest element of $\overline{\mathrm{T}_j}$.
3. Repeat step 1-3 until the number of PN sequences for the j^{th} transmitter is satisfied.

4. Update PN_j for transmission.

In this section, we have proposed two heuristic algorithms for the code assignment problem. From an initial set of PN sequences *C*, the best code sequence which minimizes the number of collisions is extracted and selected. If more than one PN sequence is available, then the PN sequence which minimizes the MAI is chosen.

4.7. Simulation and Parameters

Simulations have been conducted using the simulator OMNeT++ [VAR] and the mobility framework developed by the TKN group in Technische Universität Berlin [LOB]. 100 nodes have been simulated in a 1000x1000 m^2 field. Each node has been assigned a single PN sequence for transmission. Random initialization is used in our system; the proposed code assignment scheme is compared to the random code assignment scheme using the same set *C* of PN sequences available. We use the Gold code family of length 127 for which the average cross correlation is set equal to 0.003 (see Table 2.1). The Signal to Noise Ratio is defined by the relation

$$SNR_{\text{dB}} = 10\log\left(\frac{G.P_0.T_b}{N_0}\right) = 72.6\text{dB} + 10\log P_0 \qquad (4.23)$$

where P_0 is the common transmission power of the node, G is the reference path loss gain and $\frac{N_0}{2}$ is the AWGN variance. The received power between the i^{th} transmitter and the j^{th} receiver is defined by the relation

$$P_{i,j} = \frac{G.P_0}{d_{i,j}{}^{\alpha}} \tag{4.24}$$

where α is the path loss coefficient set equal to 3.5 and $d_{i,j}$ is the distance between the i^{th} transmitter and the j^{th} receiver. The other parameters are given in the Table 4-5.

Table 4-5- Parameters for the TOCA ad hoc network simulation

Transceiver Parameters	
Data packet size	4168 bits
Control Packet	50 bits
Data packet interval	0.2 s
Cam Interval	1 s
Data bit rate	100Kbs
Control bit rate	10Kbs
SNR threshold γ_0	6dB
Update Interval Mobility	0.5s
Delete Period	1.2 s
Cross Correlation variance	0.003
Random time t1	0.02s
Wait_For_PCAM_ACK	0.025s
Locked_Time	0.05s
T1	0.2s
T2	0.07s
T3	0.17s
T4	1s

The model used for our mobility pattern is the Constant Speed Mobility model [LOB]. In this model, the user defines a velocity for each host and an update interval; it then calculates a random target destination for the host to reach. By considering the velocity and the update frequency component it computes the number of steps and each step-size to reach the destination. At every update interval, each node computes its new position and updates the display. Once the target position is reached the node re-calculates a new random target.

4.8. Influence of the Transmission Power

The first set of simulations is aimed to show the influence of the transmission power on the performance of the proposed system. In this model, we set the mobility velocity to 5m.s^{-1} for $N = \{5,10,15,20\}$ and vary P_0 from 1mW to 5mW. The graphs are plot against the average number of neighbors captured by each node when the transmission power varies. The average number of neighbors is computed and extracted from the coloring table of each node. Its capture is not perfect and varies according to the frequency of update and the expiration time of each entry in the table. Nevertheless, the coloring table gives us a reliable indicator on the number of neighbors a node has as its transmission power changes. Figure 4-10 depicts the average number of neighbors as the transmission power varies.

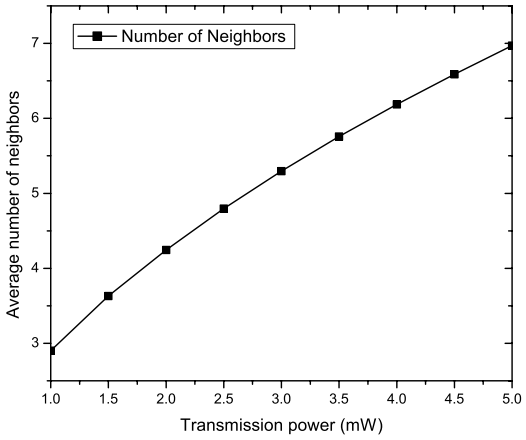
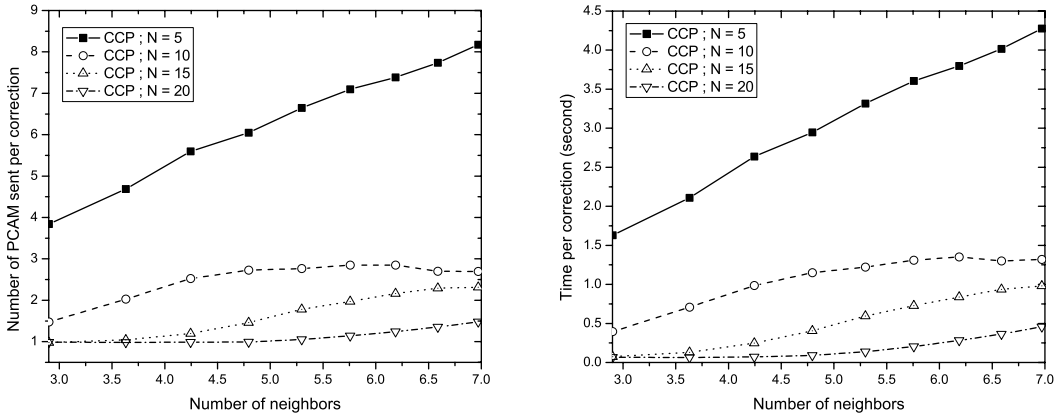


Figure 4-10-Average number of neighbors vs. Transmission power P_0

4.8.1. Correction Process

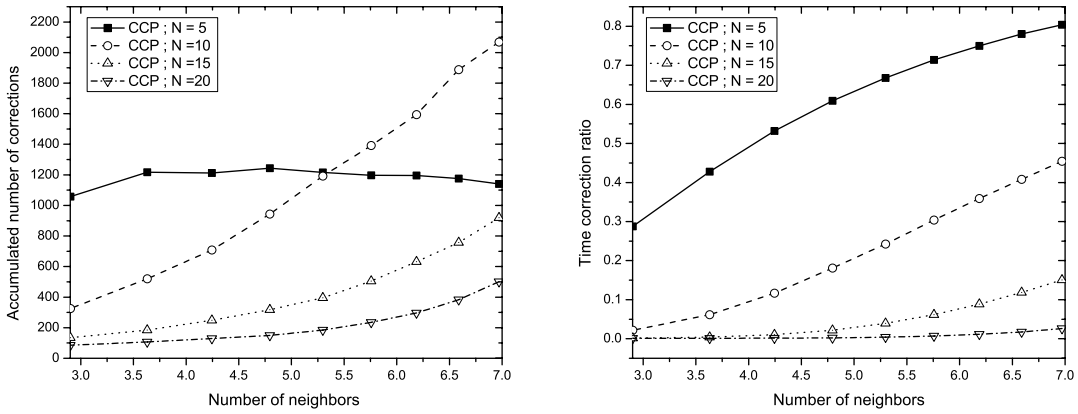


(a) (b)

Figure 4-11 - PCAM sent and time needed per correction vs. Number of neighbors Δ

Figure 4-11 describes the number of PCAM packets sent (a) and the time needed per correction (b) as the average number of neighbors Δ varies. For *N=5*, we observe that both number of PCAM packets and time per correction are very large. It is due to the fact

that the instantaneous number of violated nodes is very high. It is impossible for nodes to find an available PN sequence to correct their violation; thus, nodes retransmit PCAM packets to correct their violation without success. Nodes select at each attempt the "best code sequence" to limit the number of collisions in the network. For $N = \{10,15,20\}$, the number of PCAM packets sent per correction is relatively low and increases slowly as Δ increases. This is due to the fact that the increase of Δ also rises the number of PN sequences needed to guarantee a violation-free network, from Theorem 2 in Section 2.2.3, this number must be at least greater than Δ. The time needed for each correction follows the same trend as the number of PCAM packets sent. For Δ = 5.29, the system achieves a very reasonable correction time of 1.22s, 0.59s and 0.13s, for $N = \{10,15,20\}$, respectively.



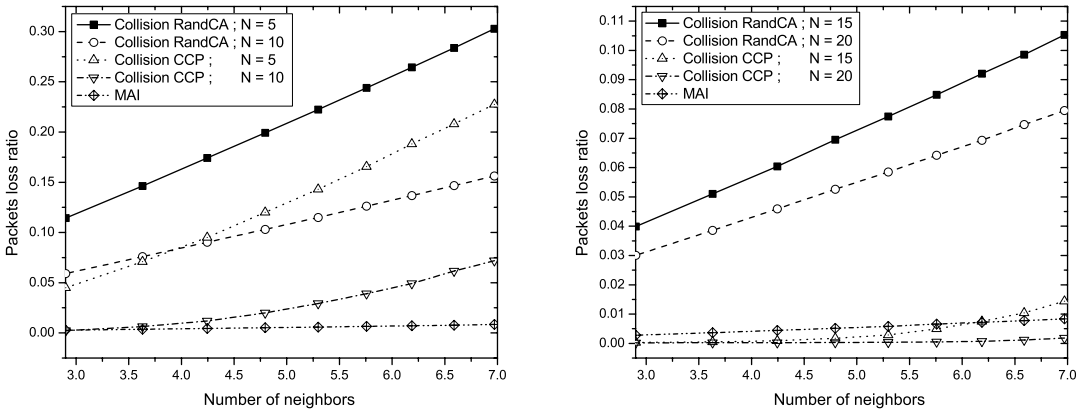
(a) (b)

Figure 4-12 –Number of corrections and time correction ratio vs. Number of neighbors Δ

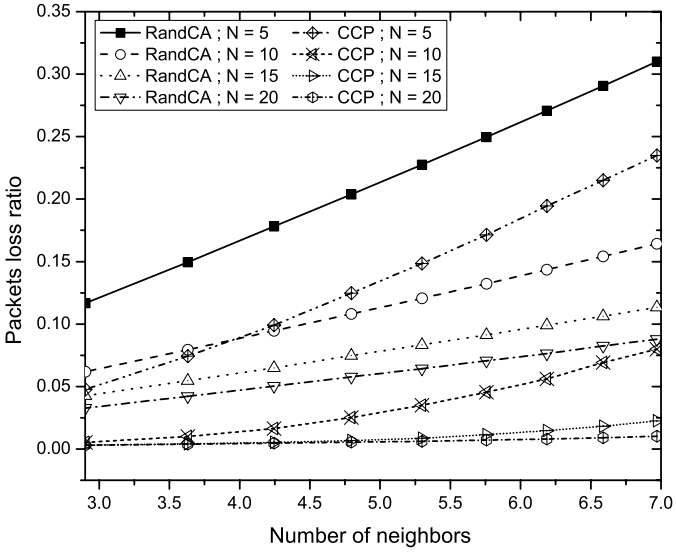
Figure 4-12 depicts the accumulated number of corrections (a) and the accumulated time correction ratio (b) for each node as Δ varies. The increase of Δ will increase the

instantaneous number of violations which will therefore increase the accumulated number of corrections needed when correction is possible ($N = \{10,15,20\}$). In parallel, the faster is a correction, the higher is achieved the number of corrections. Therefore, the accumulated number of corrections (a) behaves similarly to the time needed per correction depicted in Figure 4-11 for $N = \{10,15,20\}$. (b) represents the ratio between the accumulated time for correction and the simulation time. As we can see, this ratio increases with Δ.

4.8.2. Packets Loss and Received



(a) (b)



(c)

Figure 4-13 - Packets loss ratio vs. Number of neighbors Δ

Figure 4-13 depicts the ratio between the number of packets loss and the number of packets received as Δ varies. (a) and (b) depict the packet loss ratio due to collision and to MAI for low $N = \{5,10\}$ and high $N = \{15,20\}$. As we can see, the packets loss ratio due to collisions increases with Δ while the packets loss ratio due to MAI is stable. We

can also observe that, for low N, the packets loss ratio due to collisions dominates the packets loss due to MAI, and for high N, this domination is inverted. The plot (c) depicts the accumulated number of packets loss as Δ varies. The proposed code assignment reduces immensely the number of packets loss compared to the random code assignment. E.g. at Δ = 5.29, the proposed code assignment scheme reduces the number of packets loss by 34%, 70%, 89% and 90% over the random code assignment (RandCA presented in 2.2.1.2) schemes for $N = \{5,10,15,20\}$.

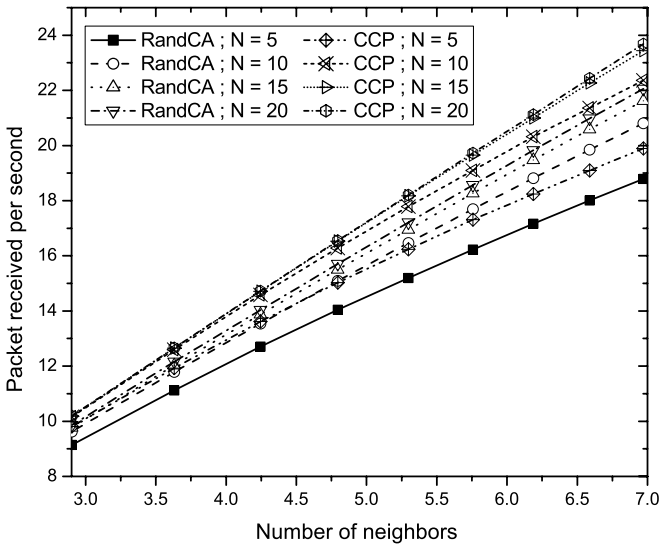


Figure 4-14 – Throughput vs Number of neighbors Δ

Figure 4-14 shows that the number of packets received per second increases steadily when Δ increases. Both the proposed code assignment and random assignment have the same behavior when Δ is low. As Δ increases, the proposed code assignment scheme has much better performances than the random code assignment scheme. At Δ = 5.29, the proposed code assignment scheme has a throughput of 17.78 packets/s for $N = 10$

and is greater than the throughput of the random code assignment scheme of 17.21 packets/s for $N = 20$. The throughputs for $N = \{15, 20\}$ of the proposed code assignment scheme essentially coincide.

We observe that, for $N = 10$, the proposed code assignment scheme has satisfactory performance for the correction process. However, the number of code re-adjustments increases considerably with Δ. This number can be reduced by increasing N which results in an increase of the mobile host design complexity.

4.9. Influence of the Velocity

The second experiment conducted aims to show the impact of mobility on the performance of the system using the proposed code assignment scheme compared to the random code assignment scheme. We vary the velocity of each node between 1m.s^{-1} and 10m.s^{-1} and set $P_0 = 3\text{mW}$.

4.9.1. Number of Neighbors

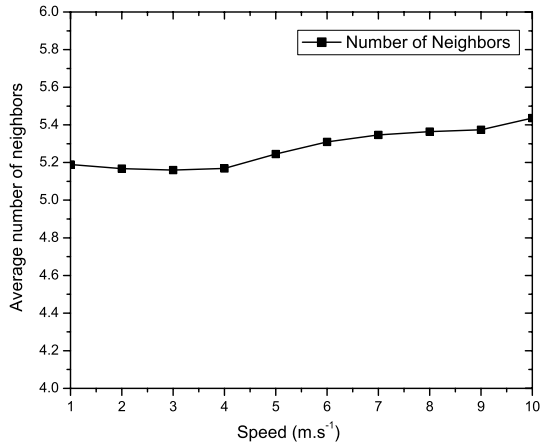
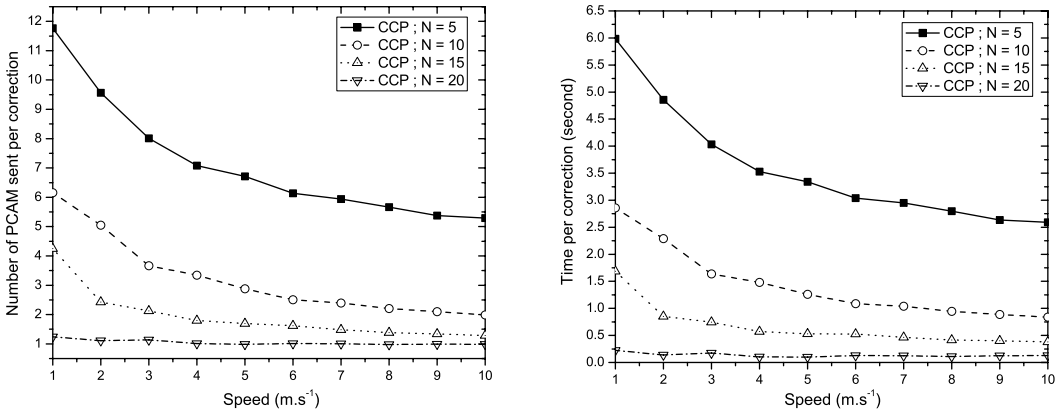


Figure 4-15 – Average number of neighbors captured at transmitter vs. Speed

Figure 4-15 shows that the average number of neighbors captured by each node increases with the velocity. The node has the illusion that it has more neighbors than it actually has. Because the expiration time of each link has been kept fixed, fast moving node might be connected and then disconnected to a node in a short period of time. Consequently, the coloring tables are less accurate. To fix this problem, the frequency of update of each link must be greater and the expiration time for each link must be reduced when the average velocity increases.

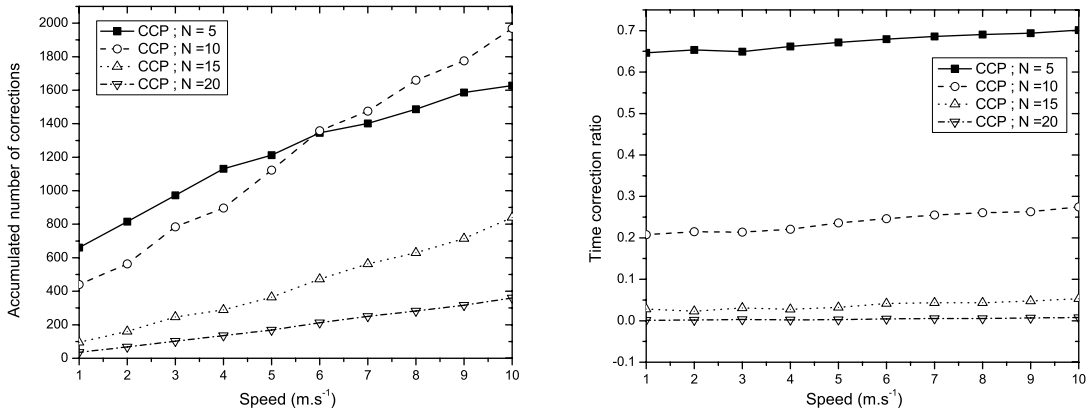
4.9.2. Correction Process



(a) (b)

Figure 4-16 - PCAM and time needed per correction vs. Speed

Figure 4-16 shows the number of PCAM packets sent and the time needed per correction as the velocity varies. We observe that nodes with higher velocities can help the correction process. When no code is available for correction, the motion of a node might correct the violation or free a PN sequence for use in the code correction process. For example, when a violation occurs and no code is anymore available, mobility can change the connectivity of nodes, this might have the effect of disconnecting two violated nodes or freeing a color for correction. This shows that the proposed protocol reacts efficiently to the frequent changes in topology.



(a) (b)

Figure 4-17 –Accumulated number of corrections and time correction ratio vs. Speed

Figure 4-17 depicts the accumulated number of corrections and the time correction ratio for each node as the velocity varies. The faster is a correction, the higher is achieved the number of corrections. Therefore, the accumulated number of corrections (a) behaves oppositely to the time needed per correction. The accumulated number of corrections and the time correction ratio of each node increase with the velocity. This is due to the fact that the topology changes more frequently and therefore nodes need more corrections.

As the speed increases, more corrections will be needed as the topology changes more frequently. In the other hand, the speed helps violated node to correct their PN sequences. The increase in the number of PN sequences available in the system will decrease the number of corrections and control packets exchanged in the system.

4.9.3. Packet Loss and Throughput

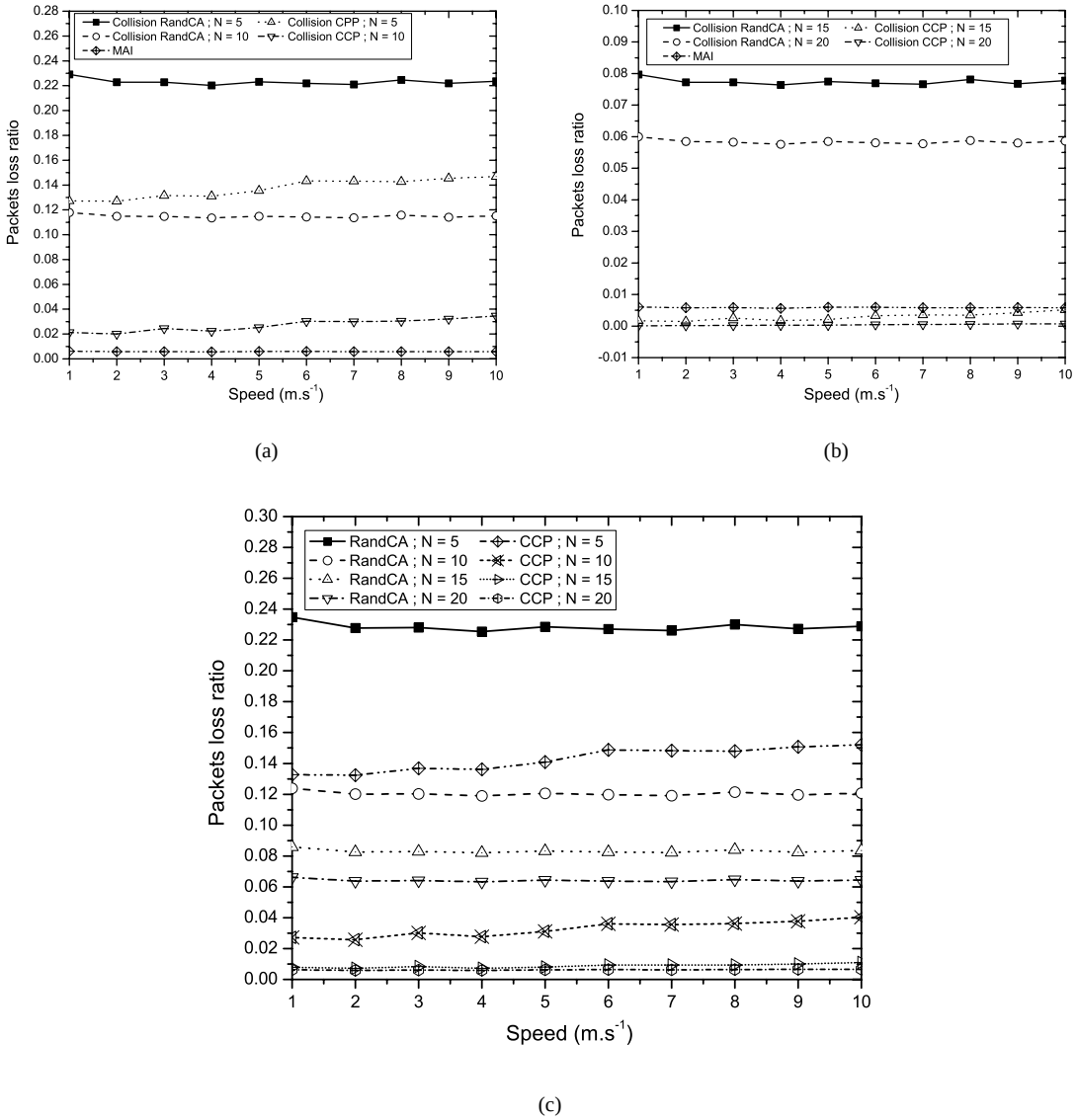


Figure 4-18 –packets loss ratio vs. Speed

Figure 4-18 depicts the ratio between the number of packets loss and the number of packets received successfully as the velocity varies. As we can see in (a) and (b), the packets loss ratio due to MAI and the packets loss ratio due to collisions for the random code assignment system are stable, while these number increases for the proposed code assignment when the velocity increases. It is due to the difficulty for nodes to obtain

accurate coloring tables for high velocity. We can also observe that, for low *N*, the packets loss ratio due to collisions dominates the packets loss due to MAI, and for high *N*, the domination is inverted.

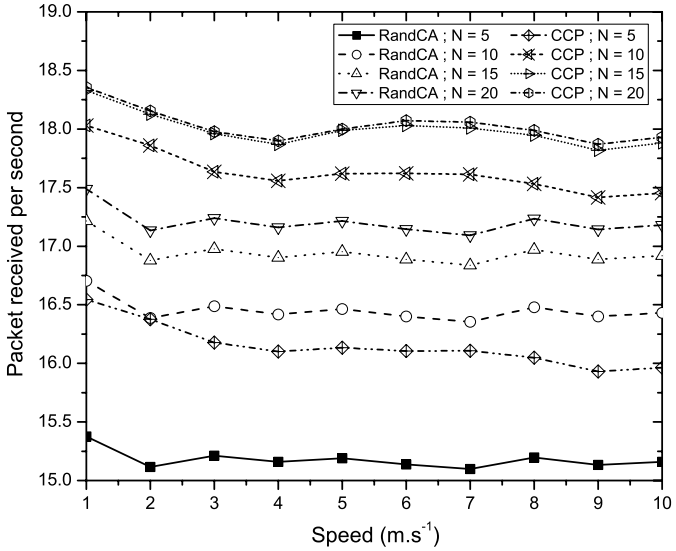


Figure 4-19 – Number of packets received per second vs. Speed

Figure 4-19 compares the number of packets received per second of the proposed code assignment scheme and the random code assignment scheme as the velocity varies. This figure reveals that the number of packets received per second using the proposed code assigned decreases slowly as the velocity increases while this number is steady for the random code assignment.

The second set of simulation has only confirmed the tradeoff created by node mobility between the hardware complexity (number of PN sequences available *N*) and the software complexity (number of control packets exchanged due to corrections).

4.10. Conclusion

In this chapter, we have demonstrated the importance of the correction process over the initialization process. We have introduced the CCP protocol and algorithms for the code assignment problem. The CCP protocol ensures that only one violating node is correcting its PN sequences at a time amongst its 2-hop neighbors including itself. When a node has locked its 1-hop neighbors using CCP, it proceeds to the correction of its violation in which the "best PN sequences" are chosen to limit collisions and MAI.

We have also demonstrated the existence of a trade off between hardware and software complexity through simulation. The set of PN sequences available must be large enough to avoid too many code sequence re-adjustments and to limit the number of control packets exchanged in the network.

With the latest progress in CDMA and multi-user detection, the adaptive multi-code assignment for ad hoc networks can become a reality. Using the same mobile device for any type of applications and networks, users will be able to be connected to a third generation network (WCDMA or CDMA2000) and switch to an ad hoc network or a WLAN spot when available by using the proposed multi-code assignment scheme.

Chapter 5 Implementation of a TOCA Mobile Ad Hoc Network with OMNeT++

This Chapter presents the implementation of the CCP protocol and the code assignment algorithm proposed for a real time mobile ad hoc network using OMNeT++.

Section 5.1 gives a short introduction of OMNeT++ as well as the mobility framework used for our simulation. Section 5.2 then presents the design of the mobile host for a TOCA system. Finally, Section 5.3 describes the channel control used and analyses the complexity of the simulator.

5.1. OMNeT++ and Mobility Framework

OMNeT++ is an event simulator developed in the University of Budapest by Varga [VAR]. It is an open-source software entirely coded in C++ and used for an graphical interface a tcl/tk environment. In 2004, the University of Berlin introduced to OMNeT++ programmers a new framework for wireless communication [LOB] which can be used to simulate ad hoc or centralized access point networks.

5.2. The mobile Host

We modified the mobile host structure from [LOB] by introducing the Color Module which is implemented at the MAC layer. Figure 5-1 depicts the design of the mobile host in OMNeT++.

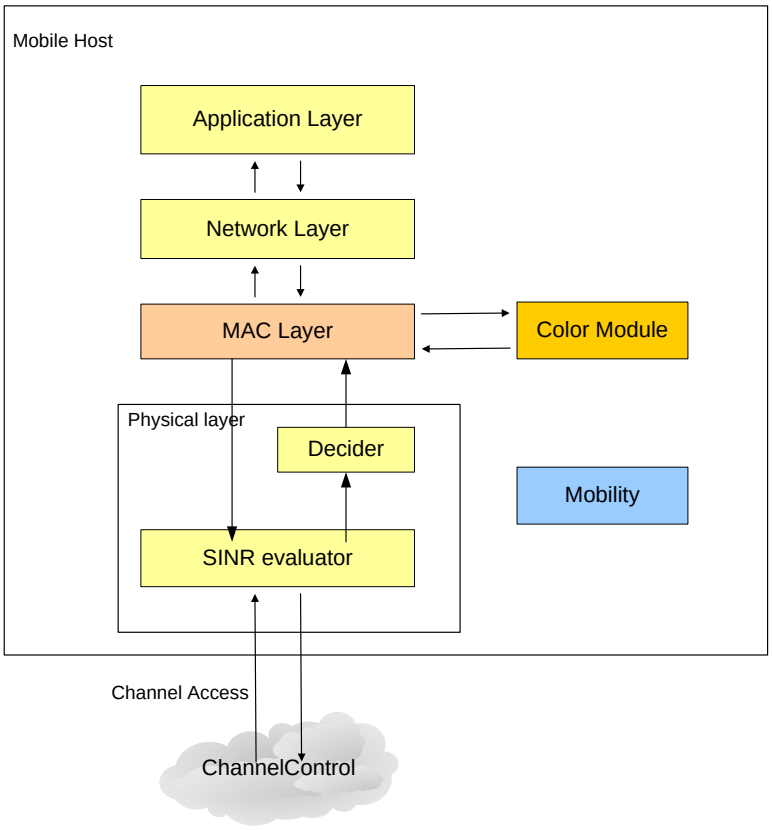


Figure 5-1 Design of the mobile host in OMNeT++

5.2.1. The Physical Layer

The physical layer is divided into two layers: the SINR evaluator and the Decider layers.

5.2.1.1. The SINR Evaluator

For reception, the SINR evaluator is used to simulate the transmission delay and calculate the different SINR levels associated to a received packet. When a packet *p* is received, it is held in the container *MessageBuffer* during its time duration expressed as $\partial_p = \frac{sizePacket(p)}{bitRate}$, where *sizePacket(p)* denotes the size of the packet *p* and *bitRate* is the bit rate of the channel. For each packet *p*, it is associated a list *SINRlist(p)* which stores the different SINR levels computed during its time duration. At any time a new packet is received, the SNR evaluator re-computes the SINR levels of the packets contained in *MessageBuffer*. The SINR level of each packet *p* is given by

$$SINR_{level}(p) = \frac{P(p)}{\sum_{p' \in MessageBuffer - \{p\}} cc \times P(p') + \frac{N_0}{T_b}} \tag{5.1}$$

where *P(p)* is the received power associated to the packet *p* and *cc* is defined as the average value of the elements of the cross-correlation matrix *CC* defined in Section 4.6.1.

When the packet *p* is received from a channel already used by another packet *p'* contained in *MessageBuffer* with an initial $SINR_{level}(p') \geq \gamma_0$ and if its initial SINR level satisfies $SINR_{level}(p) \geq \gamma_0$, then the packet *p* is said to be collided.

When the time duration of the packet *p* in *MessageBuffer* has expired, the packet *p* which does not satisfy $\frac{P(p)}{N_0} \geq \gamma_0$ or is collided is deleted at the SINR evaluator. Otherwise, the packet *p* is sent to the Decider with its *SINRlist(p).*

For transmission, when a packet *p* is received from the MAC layer, the SINR evaluator puts the packet *p* in a transmission buffer and waits for each packet p' which is currently in *MessageBuffer* with an initial $SINR_{level}(p') \geq \gamma_0$ to be completely received. The packet *p* is then un-buffered and broadcasted using the PN sequence *pn(p)* associated to the packet *p* by the MAC layer to all its interfering neighbors. Node A and node B are said to be two interfering neighbors when their distance is at most the maximum inference distance $d_{\text{maxinterference}}$ given by

$$d_{\text{maxinterference}} = \left[\frac{P_0.G}{10^{sat/10}}\right]^{1/\alpha} \tag{5.2}$$

where $P_0, G,$ and α are constant defined in Section 4.7 and *sat* is the saturation coefficient of the receiver and it is set equal to -110dB.

5.2.1.2. The Decider

When the packet *p* is received from the SNR evaluator, the decider extracts its associated *SINRlist(p)*. It verifies that all the $SINR_{level}(p)$ from *SINRlist(p)* are greater than γ_0. If

this condition is satisfied, the packet *p* is then sent to the MAC layer; otherwise, the packet is said to be corrupted by MAI and subsequently deleted.

5.2.2. The MAC Layer

At the MAC layer, when a data packet is received, the MAC layer compares the packet's destination and its address; if the address matched then the packet is sent to the Network layer otherwise it is deleted. When a control packet is received, it is sent for processing to the Color module following the CCP protocols presented in Section 4.4.

When a packet *p* is received from the Network layer, the MAC layer decides with which PN sequences *pn(p)* the packet *p* will be transmitted. The packet *p* with its PN sequence information *pn(p)* is then sent to the SINR evaluator level for transmission.

5.2.3. The Network and the Application Layer

In our work, we have focused our interest on the MAC and the Physical layers. Therefore, the Network and Application layers have been implemented with only elementary functions. The Network layer sends packets to its lower and upper layers and the Application layer only generates data packets following an exponential law, with mean μ and counts the number of data packets successfully received.

5.3. The Channel Control

The *ChannelControl* is responsible for connecting nodes within a distance lower than the $d_{\text{maxinterference}}$ and disconnecting them when their distance is greater than $d_{\text{maxinterference}}$. At any motion, the system needs to update the connections between hosts. The system computes the distance between every pair of the *K* hosts in the network. This operation has a complexity of $O(K^2)$.

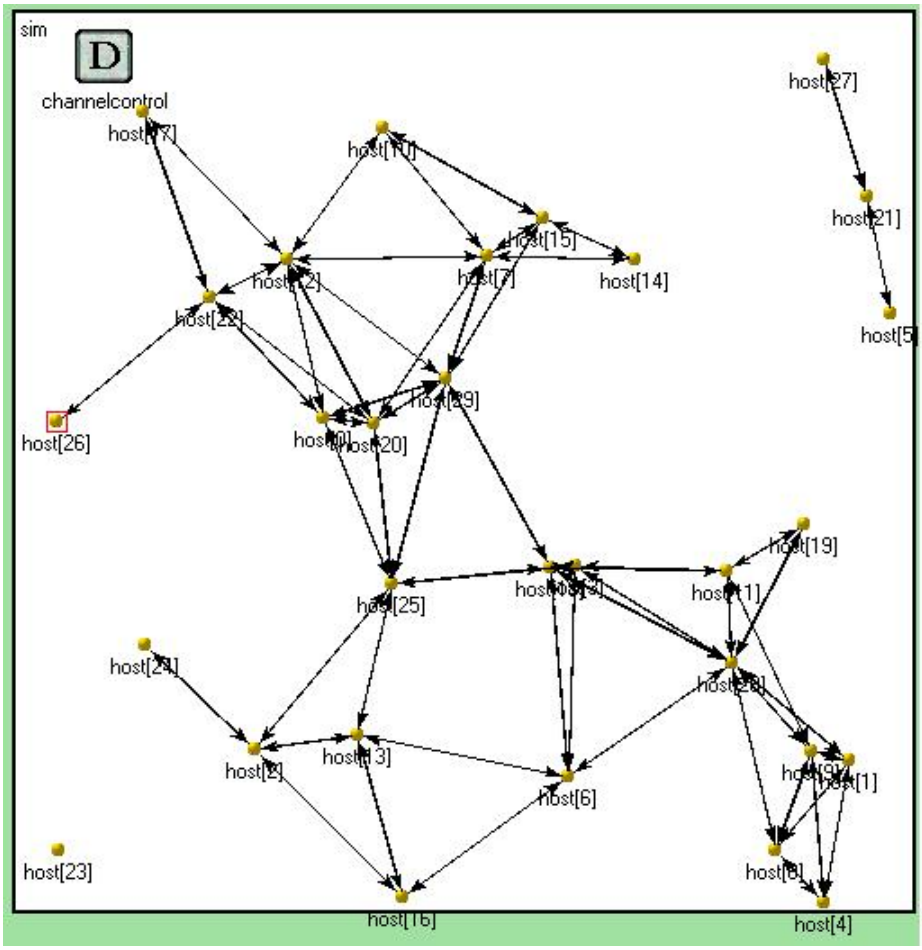


Figure 5-2- OMNeT++ with 30 nodes

Chapter 6 Conclusion

In this thesis, we introduced new multi-code assignment protocols and algorithms for DS-CDMA ad hoc networks. We analyzed and compared different strategies used for a code assignment scheme and showed that only TOCA and POCA can completely eliminate the hidden terminal problem. We also highlight the importance of the code correction process over the initialization process for a code assignment scheme. We have as well demonstrated the impact of the number of PN sequences available in the system on the performances of the code assignment scheme.

In our first contribution, we have presented a new multi code assignment scheme for small sized POCA networks which is able to combine code assignment and power control together to prevent collisions and to mitigate MAI and fading.

In our second contribution, we have presented a new distributed and adaptive code assignment protocol for TOCA ad hoc networks. Each violating node, with the CCP protocol, utilizes their 1-hop neighbors as locker avoiding that one of its 2-hop neighbors is correcting simultaneously its PN sequences. When the locking process is successful, the violating node moves to the correction of its PN sequences by using the proposed

code assignment algorithm which selects the “best code sequences” to limit collisions and MAI. An analysis of the proposed scheme as well as a discussion on the impact of the transmission power and the mobility are given. A tradeoff created by node mobility between the hardware complexity (number of PN sequences available) and the software complexity (number of control packets exchanged due to corrections) has also been demonstrated.

In the last contribution, we have implemented the first real time simulator for TOCA mobile ad hoc networks for OMNeT++.

Mobility has emphasized the difficulty for nodes to obtain accurate tables. This problem could be included and discussed in the first orientation of our future work. Also, the code assignment scheme which assigns for each transmitter the “best PN sequences” may result in 2-hop neighbors sharing identical PN sequences for transmission when no more code is available for correction. In the second orientation, to provide better throughput, a node may then adapt its arrival rate of data packet with the number of 2-hop neighbors which are sharing the same PN sequences [SHA03, YUE02]. In the third orientation, the roaming between TOCA ad hoc networks and other existing networks may also be studied.

Reference

[ABR70] N. Abramson, “The ALOHA System – Another Alternative for Computer Communications,” *Proc. The Full Joint Computer Conference*, pp. 281-285, 1970

[ABR73] N. Abramson, “Packet Switching with Satellites,” *AFIPS Conf. Proc., NCC*, pp. 695-702, 1973

[ABR77] N. Abramson, “The Throughput of Packet Broadcasting Channels,” *IEEE Trans. Commun.* , Vol. COM-25, No. 1, pp. 117-128, January 1977

[ABR94] N. Abramson, “Multiple Access in Wireless Digital Networks,” *Proc. IEEE*, Vol. 82, pp. 1360-1369, 1994

[BER95] A.A. Bertossi, M.A. Bonuccelli, “Code Assignment for Hidden Terminal Interference Avoidance in Multihop Packet Radio Networks,” *IEEE/ACM Transactions on Networking*, Vol. 3, Issue 4, pp. 441-449, 1995

[BAT99] R. Battiti, A. A. Bertossi, M. A. Bonuccelli, “Assigning Code in Wireless Networks: Bounds and Scaling Properties,” *Wireless Networks*, vol.5 pp 195-209. 1999

[BAT00] R. Battiti, A. A. Bertossi, M. A. Bonuccelli, “Distributed Code Assignment in Multihop Radio Networks: Object-Oriented Software Simulations,” *Proc. of SoftCOM Rijeka*, 2000

[CHL03] I. Chlamtac, M. Conti, J. Liu, "Mobile Ad hoc Networking: Imperatives and Challenges," *Ad Hoc Network Journal*, Vol.1, No.1, January-February-March 2003.

[CID89] I. Cidon, M. Sidi, "Distributed Assignment Algorithms for Multihop Packet Radio Networks Computers," *IEEE Trans. Comput.*, vol. 38, pp. 1353-1361, 1989

[FUL98] C. L. Fullmer, "Collision Avoidance Techniques for Packet-Radio Networks," *PhD Thesis*, University of California, 1998

[GAR97] J.J. Garcia-Luna-Aceves, J. Raju, "Distributed Assignment of Codes for Multihop Packet-Radio Networks," *IEEE MILCOM 97 Proc.*, Vol. 1, pp. 450-454, 1997

[HU93] L. Hu, "Distributed Code Assignments for CDMA Packet Radio Networks," *IEEE/ACM Transactions on Networking*, Vol. 1 pp. 668-77, 1993

[HUN92] K.-W Hung, T.-S. Yum, "Efficient Spreading Code Assignment Algorithm for Packet Radio Networks," *Electronics Letters*, Vol. 28 pp. 2193-2195, 1992

[KAR92] K. H. A. Kärkkäinen "Mean-Square Cross-correlation as a Performance Measure for Spreading Code Families," *IEEE Second International Symposium on Spread Spectrum Techniques and Applications* (ISSSTA'92), 1992

[KIM91] D.-I Kim, R.A Scholtz, "A Random Spreading Code Assignment Scheme for Centralized Spread-Spectrum Packet Radio Networks," *IEEE MILCOM '91 Proc.*, vol.1, pp. 132-136, 1991

[LIN97] C. R. Lin, M. Gerla, "Adaptive Clustering for Mobile Wireless Networks," *IEEE Journal on Selected Areas in Communications*, Vol. 15, No. 7, pp.1265-1275, 1997.

[LOB] M. Lobbers, D. Willkomm, A. Kopke, H. Karl, "Mobility Framework," *http://www.tkn.tu-berlin.de/research/research_texte/framework.html*

[LOW03] B. Low, F. Cai, M.A. Armand, M. Motani, "Distributed Code Assignment for DS-CDMA Ad Hoc Network" *conference IEEE Digital Signal Processing and Digital Communication*, 2003.

[MIS92] J. Misra and D. Gries, “A constructive Proof of Vizing's Theorem,” *Inf. Proc. Lett.*, Issue 41, pp.131-133, 1992

[MAK87] T. Makansi, “Transmitter-Oriented Code Assignment for Multihop Radio Networks,” *IEEE Trans. Commun.*, Vol.35, No. 12, pp. 1379–82, 1987

[MAL01] N. Malpani, N. H. Vaidya, J. L. Welch, “Distributed Token Circulation in Mobile Ad Hoc Networks,” *Proc. 9th Intl. Conf. on Network Protocols*, 2001

[NAK00] K. Nakano, S. Olariu “Randomized Initialization Protocols for Ad Hoc Networks,” *IEEE trans. on Parallel and Distributed Systems*, Vol. 11, No. 7, 2000

[NAK02] K. Nakano, S. Olariu, “A Survey on Leader Election Protocols for Radio Networks,” *Proc. International Symposium on Parallel Architectures, Algorithms and Networks*, 2002

[PRO02] J. Proakis, “Digital Communication,” *Fourth Edition McGraw-Hill International*, 2002

[PUR77-1] M. B. Pursley, “Performance Evaluation for Phase-Coded Spread Spectrum Multiple-Access Communication-Part I: System Analysis,” *IEEE Trans. Commun.*, Vol. COM-25, No. 8, pp. 795-799, 1977

[PUR77-2] M. B. Pursley, “Performance Evaluation for Phase-Coded Spread Spectrum Multiple-Access Communication-Part II: Code Sequence Analysis,” *IEEE Trans. Commun.*, vol. COM-25, No. 8, pp. 800-803, 1977

[PUR87] M. B. Pursley, “The Role of Spread Spectrum in Packet Radio Networks,” *Proc. IEEE*, 1987

[RAM99] S. Ramanathan, "A Unified Framework and Algorithm for Channel Assignment in Wireless Networks," *Proc. IEEE INFOCOM*, 1997

[RAP02] T. S. Rappaport, "Wireless Communications", Principles and Practice," *Second Edition, Prentice Hall International Editions,* 2002

[ROM01] K. Romer, "Time Synchronization in Ad Hoc Networks," *Proc. 2nd ACM international symposium on Mobile ad hoc networking & computing*, 2001

[RUB02] I. Rubin, A. Behzad, R. Zhang H. Luo, E. Caballero "TBONE: A Mobile-Backbone Protocol for Ad Hoc Wireless Networks," *IEEE Aerospace Conference Proc.* Vol. 6, pp. 2727-2740, 2002

[SEK99] T. Seki, M. Hamamura, S. Tachikawa, "Suppression Effects of Multiple Access Interference in DS/CDMA with Code-Diversity," *IEICE Trans. Fundamentals*, Vol. E82-A, No.12, pp. 2720-2727, 1999

[SCH77] R. A. Scholtz "The Spread Spectrum Concept," *IEEE Trans. Commun.*, Vol. COM-25, no. 8, pp. 748-755, .1977

[SHA03] S. Shakkottai, T.S. Rappaport P. C. Karlsson, "Cross-Layer Design for Wireless Networks", *IEEE Communications Magazine*, Vol. 41, No. 10, pp. 74 – 80, 2003

[SOU88] E.S. Sousa, J.A Silvestre, "Spreading code protocols for distributed spread-spectrum packet radio networks", *IEEE trans. Commun.*, pp 272-281, 1988

[TOG75-1] F.A Tobagi, L. Kleinrock, "Packet Switching in Radio Channels: Part I – Carrier Sense Multiple Access Modes and their Throughput-Delay Characteristics," *IEEE Trans. Commun.*, Vol. COM-23, No. 12, pp. 1400-1416, 1975

[TOG75-2] F.A Tobagi and L. Kleinrock. "Packet Switching in Radio Channels: Part II – the Hidden Terminal problem in Carrier Sense Multiple Access Modes and the busy tone solution," *IEEE Trans. Commun.*, Vol. COM-23 No.12, pp. 1417-1433, 1975

[VAR] A. Varga, "OMNeT++ discrete event simulator system," http://www.omnetpp.org/

[WU02] Y. Wu, Q. Zhang, W. Zhu, S.-Y. Kung, "Spreading Code Assignment in an Ad-Hoc DS-CDMA Wireless Network," *International Conference on Communications (ICC2002)*, 2002

[YAO77] K. Yao "Error Probability of Asynchronous Spread Spectrum Multiple Access Communication Systems," *IEEE Trans. Commun.* Vol. COM-25 pp. 803-809, Aug. 1977

[YUE02] W. H. Yuen, H. Lee, T. D. Andersen, "A Simple and Effective Cross Layer Networking System for Mobile Ad Hoc Networks," *Proc. IEEE PIMRC*, 2002

Appendix A – Derivation of Average SINR for an Asynchronous DS-CDMA System

Appendix A derives a simple expression of the average SINR for an asynchronous DS-CDMA system presented in chapter 2. We present some results extracted from [PUR77-1, PUR77-2].

We consider *K* simultaneous transmitters. The k^{th} transmitter is assigned a PN sequence for transmission $a_k(t)$ as defined in Section 2.1. The signals from the K transmitters arrive simultaneously at the reference receiver 0. At the receiver 0, the received signal is given by

$$\begin{aligned} r_0(t) &= \sum_{k=1}^{K} s_{k,0}(t-\tau_k) + n(t) \\ &= \sum_{k=1}^{K} \sqrt{P_{k,0}}\, b_k(t-\tau_k) a_k(t-\tau_k) \cos(\omega_c t + \phi_k) + n(t) \end{aligned} \tag{A.1}$$

where $n(t)$, $s_{k,0}(t)$, $P_{k,0}$, $b_k(t)$, $a_k(t)$, τ_k, and ϕ_k are defined in Section 2.1.

At the reference receiver 0, the received signal is multiply by the code sequence of the desired user and then integrated over one bit period T_b. Assuming that the desired user is the 1st transmitter for example and that the reference receiver 0 is delay and phase synchronizes with the desired user ($\tau_1 = 0$ and $\phi_1 = 0$). Thus we can obtain the static decision for the transmitted bit $b_1(t)$ from the 1st transmitter as

$$Z_{0,1,a_1} = \int_0^{T_b} r_0(t) a_1(t) \cos(\omega_c t) dt \tag{A.2}$$

Substituting (A.1) in (A.2), the static decision for the transmitted bit $b_1(t)$ can be obtained as

$$\begin{aligned} Z_{0,1,a_1} &= \int_0^{T_b} \left[\sum_{k=1}^{K} \sqrt{P_{k,0}} b_k(t-\tau_k) a_k(t-\tau_k) \cos(\omega_c t + \phi_k) + n(t) \right] a_1(t) \cos(\omega_c t) dt \\ Z_{0,1,a_1} &= \int_0^{T_b} \sqrt{P_{1,0}} . b_1(t) . a_1^2(t) . \cos^2(\omega_c t) dt \\ &\quad + \sum_{k=2}^{K} \sqrt{P_{k,0}} . \int_0^{T_b} b_k(t-\tau_k) . a_k(t-\tau_k) . a_1(t) \cos(\omega_c t + \phi_k) \cos(\omega_c t) dt \\ &\quad + \int_0^{T_b} n(t) a_1(t) \cos(\omega_c t) dt \end{aligned} \tag{A.3}$$

We can extract from (A.3), the desired signal component $D_{0,1,a_1}$ of the 1st transmitter, the multiple access interference (MAI) component $MAI_{0,1,a1}$, and the thermal noise component η. The static decision can be written as

$$Z_{0,1,a_1} = D_{0,1,a_1} + MAI_{0,1,a_1} + \eta \tag{A.4}$$

The desired signal component is given by

$$\begin{aligned} D_{0,1,a_1} &= \int_0^{T_b} \sqrt{P_{1,0}}.b_1(t).a_1^2(t).\cos^2(\omega_c t)dt \\ &= \sqrt{P_{1,0}}.b_1(0).\int_0^{T_b} \cos^2(\omega_c t)dt \\ &= \frac{1}{2}\sqrt{P_{1,0}}.b_1(0).\int_0^{T_b} \left(1+\cos(2\omega_c t)\right)dt \\ &\approx \sqrt{P_{1,0}}.b_1(0).\frac{T_b}{2} \qquad \text{for } \omega_c \gg \frac{1}{T_b} \end{aligned} \tag{A.5}$$

The thermal noise η is expressed as

$$\eta = \int_0^{T_b} n(t)a_1(t)\cos(\omega_c t)dt \tag{A.6}$$

from (A.6), we can derive the mean of η as

$$\begin{aligned} \mathrm{E}[\eta] &= \mathrm{E}\left[\int_0^{T_b} n(t)a_1(t)\cos(\omega_c t)dt\right] \\ &= \int_0^{T_b} \mathrm{E}[n(t)]a_1(t)\cos(\omega_c t)dt = 0 \end{aligned} \tag{A.7}$$

where $\mathrm{E}[n(t)] = 0$.

The variance of η is given by

$$\begin{aligned} Var[\eta] &= \mathrm{E}[\eta^2] - (\mathrm{E}[\eta])^2 = \mathrm{E}[\eta^2] \\ &= \mathrm{E}\left[\int_{t=0}^{T_b}\int_{u=0}^{T_b} a_1(u).a_1(t).n(t).n(u)\cos(\omega_c t)\cos(\omega_c u)du.dt\right] \\ &= \int_{t=0}^{T_b}\int_{u=0}^{T_b} a_1(u).a_1(t).\mathrm{E}[n(t).n(u)]\cos(\omega_c t)\cos(\omega_c u)du.dt \\ &= \int_{t=0}^{T_b}\int_{u=0}^{T_b} a_1(u).a_1(t)\frac{N_0}{2}\delta(t-u)\cos(\omega_c t)\cos(\omega_c u)du.dt \\ &= \int_{t=0}^{T_b} a_1^2(t)\frac{N_0}{2}\cos^2(\omega_c t)dt = \frac{N_0}{4}\int_{t=0}^{T_b}(1+\cos(2\omega_c t))dt \\ &\approx \frac{N_0 T_b}{4} \qquad \text{for } \omega_c \gg \frac{1}{T_b} \end{aligned} \tag{A.8}$$

where $\delta(t)$ is the Dirac function which takes the value 1 for t=0 and 0 elsewhere.

We are now considering the MAI component which can be written as the sum of interference created at the reference receiver 0 matched with the 1st transmitter with the PN sequence $a_1(t)$ by each of the k^{th} transmitter using the PN sequence $a_k(t)$ for k from 2 to K is given by

$$MAI_{0,1,a_1} = \sum_{k=2\ldots K} \Phi_{0,1,a_1,k,a_k} \tag{A.9}$$

The interference created by the k^{th} transmitter $\Phi_{0,1,a_1,k,a_k}$ is given by

$$\Phi_{0,1,a_1,k,a_k} = \int_{t=0}^{T_b} \sqrt{P_{k,0}} b_k(t-\tau_k) a_k(t-\tau_k) a_1(t) \cos(\omega_c t + \phi_k) \cos(\omega_c t) dt \qquad \text{(A.10)}$$

Due to the fact that the chips $b_k(t)$ and $a_k(t)$ are rectangular and that $\omega_c \gg \frac{1}{T_b}$, we can write

$$\begin{aligned} \Phi_{0,1,a_1,k,a_k} &= \int_{t=0}^{T_b} \sqrt{P_{k,0}} b_k(t-\tau_k) a_k(t-\tau_k) a_1(t) \cos(\omega_c t + \phi_k) \cos(\omega_c t) dt \\ &= \int_{t=0}^{T_b} \sqrt{P_{k,0}} b_k(t-\tau_k) a_k(t-\tau_k) a_1(t) \left(\frac{\cos(2\omega_c t + \phi_k) + \cos(\phi_k)}{2} \right) dt \\ &\approx \frac{1}{2} \sqrt{P_{k,0}} \cos(\phi_k) \int_{t=0}^{T_b} b_k(t-\tau_k) a_k(t-\tau_k) a_1(t) dt \end{aligned} \qquad \text{(A.11)}$$

By introducing the two elements $l_k \in \{0,...,M\text{-}1\}$ and $\varepsilon_k \in [0, T_c[$, which satisfy the condition $\tau_k = l_k T_c + \varepsilon_k$, we can extract the Figure A-1 which represents the relative delay between the received signals from the k^{th} transmitter and the 1^{st} transmitter.

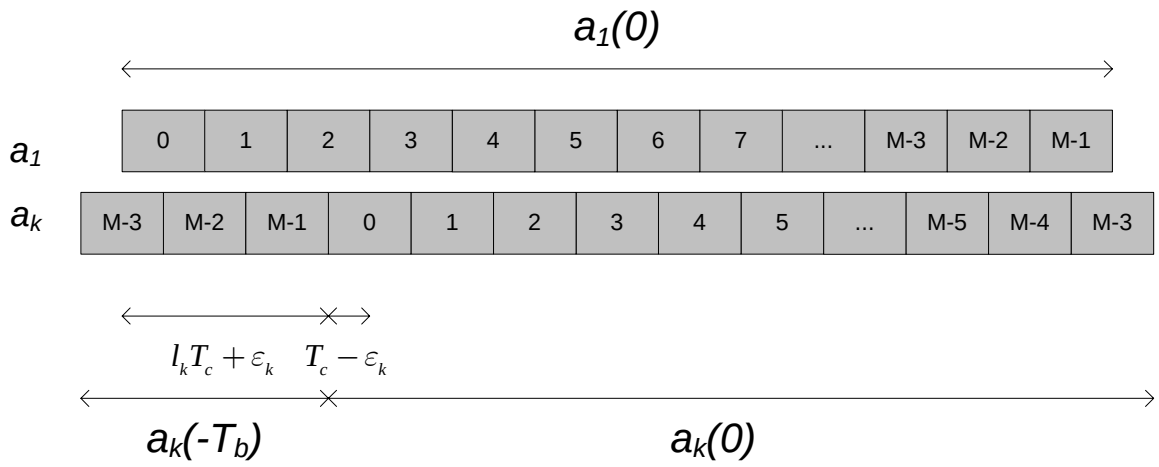


Figure A-1- Relative delay between the received signals from the k^{th} and the 1st transmitters

From Figure A-1 and (A.11) we can then write

$$\begin{aligned}\Phi_{0,1,a_1,k,a_k} = \frac{1}{2}\sqrt{P_{k,0}}\cos(\phi_k)\Bigg[& b_k(0).\sum_{j=0}^{M-l_k-2}\left[(T_c-\varepsilon_k).a_{k,j}.a_{1,l_k+j}+\varepsilon_k.a_{k,j}.a_{1,l_k+j+1}\right] \\ & + b_k(0).(T_c-\varepsilon_k).a_{k,M-l_k-1}.a_{1,M-1} + b_k(-T_b)\varepsilon_k.a_{k,M-l_k-1}.a_{1,0} \\ & + b_k(-T_b).\sum_{j=M-l_k}^{M-1}\left[(T_c-\varepsilon_k).a_{k,j}.a_{1,j-M+l_k}+\varepsilon_k.a_{k,j}.a_{1,j+1-M+l_k}\right]\Bigg] \quad \text{(A.12)} \\ = \frac{1}{2}.\sqrt{P_{k,0}}.\cos(\phi_k)\Bigg[& b_k(0).\Bigg[(T_c-\varepsilon_k).\sum_{j=0}^{M-l_k-1}a_{k,j}.a_{1,l_k+j}+\varepsilon_k.\sum_{j=0}^{M-l_k-2}a_{k,j}.a_{1,l_k+j+1}\Bigg] \\ & + b_k(-T_b).\Bigg[(T_c-\varepsilon_k)\sum_{j=M-l_k}^{M-1}a_{k,j}.a_{1,j-M+l_k}+\varepsilon_k.\sum_{j=M-l_k-1}^{M-1}a_{k,j}.a_{1,j+1-M+l_k}\Bigg]\Bigg]\end{aligned}$$

[PUR77-1, PUR77-2] defined functions to simplify the expression of the $\Phi_{0,1,a_1,k,a_k}$ which are given by

i. The discrete aperiodic cross-correlation function of the sequences $a_k = \left\{a_{k,j}\right\}_{j=0}^{M-1}$ is defined by

$$C_{a_k,a_i}(l) = \begin{cases} \sum_{j=0}^{M-1-i} a_{k,j}.a_{i,j+l} & 0 \le l \le N-1 \\ \sum_{j=0}^{M-1+i} a_{k,j-l}.a_{i,j} & 1-N \le l < 0 \\ 0, & \text{elsewhere} \end{cases} \quad \text{(A.13)}$$

ii. The even periodic cross-correlation function of the sequences $a_k = \left\{a_{k,j}\right\}_{j=0}^{M-1}$ is defined by :

$$\theta_{a_k,a_i}(l) = \sum_{j=0}^{M-1} a_{k,j}.a_{i,j+l} = C_{a_k,a_i}(l) + C_{a_k,a_i}(l-N) \quad \text{(A.14)}$$

iii. The odd periodic cross-correlation function of the sequences $a_k = \left\{a_{k,j}\right\}_{j=0}^{M-1}$ is defined by

$$\tilde{\theta}_{a_k,a_i}(l) = C_{a_k,a_i}(l) - C_{a_k,a_i}(l-N) \quad \text{(A.15)}$$

iv. The cross-correlation parameters $\mu_{a_k,a_i}(l)$ is defined by

$$\mu_{a_k,a_i}(l) = \sum_{j=1-N}^{N-1} C_{a_k,a_i}(j) C_{a_k,a_i}(j+l) \tag{A.16}$$

v. the average cross correlation parameters r_{a_k,a_i} is given by

$$r_{a_k,a_j} = 2.\mu_{a_k,a_j}(0) + \mu_{a_k,a_j}(1) \tag{A.17}$$

From (A.12) and (A.13), we can derive the expression

$$\begin{aligned}\Phi_{0,1,a_1,k,a_k} = \frac{1}{2}.\sqrt{P_{k,0}}.\cos(\phi_k).\Big[b_k(0).\Big[(Ts-\varepsilon_k).C_{k,1}(l_k)+\varepsilon_k.C_{k,1}(l_k+1)\Big] \\ +b_k(-T_b).\Big[(Ts-\varepsilon_k).C_{k,1}(l_k-M)+\varepsilon_k.C_{k,1}(l_k+1-M)\Big]\Big]\end{aligned} \tag{A.18}$$

We assume now that the $b_k(nT_b)$ are random variable identically distributed on $\{+1,-1\}$ and independents, we have then the relations

$$\begin{cases} E\left[b_k(nT_b)\right] = 0 & \forall n \in \mathbb{Z} \\ E\left[b_k(nT_b)b_k(mT_b)\right] = 0 & \forall n \neq m \in \mathbb{Z} \\ E\left[b_k(nT_b)^2\right] = 1 & \forall n \in \mathbb{Z} \end{cases} \tag{A.19}$$

We can write using (A.19)

$$
\begin{aligned}
Var\left[\Phi_{0,1,a_1,k,a_k}\,\middle|\phi_k\right] &= \frac{1}{T_b}\left[\sqrt{P_{k,0}}\,\frac{1}{2}.\cos(\phi_k)\right]^2 \sum_{l_k=0}^{M-1}\int_{\varepsilon_k=0}^{T_c}\left[\begin{bmatrix}(T_c-\varepsilon_k).C_{a_k,a_1}(l_k)\\ +\varepsilon_k.C_{a_k,a_1}(l_k+1)\end{bmatrix}^2 + \begin{bmatrix}(Ts-\varepsilon_k).C_{a_k,a_1}(l_k-M)\\ +\varepsilon_k.C_{a_k,a_1}(l_k+1-M)\end{bmatrix}^2\right] d\varepsilon_k \\
&= \frac{1}{T_b}\frac{1}{4}.P_{k,0}.\cos^2(\phi_k).\frac{T_c^3}{3}.\sum_{l_k=0}^{M-1}\begin{bmatrix}C_{a_k,a_1}(l_k)^2 + C_{a_k,a_1}(l_k+1)^2\\ +C_{a_k,a_1}(l_k-M)^2 + C_{a_k,a_1}(l_k+1-M)^2\\ +C_{a_k,a_1}(l_k)C_{a_k,a_1}(l_k+1)\\ +C_{a_k,a_1}(l_k-M)C_{a_k,a_1}(l_k+1-M)\end{bmatrix} \\
&= \frac{1}{T_b}\frac{1}{12}.P_{k,0}.\cos^2(\phi_k).T_c^3.\left[2.\mu_{a_k,a_1}(0)+\mu_{a_k,a_1}(1)\right] \\
&= \frac{1}{T_b}\frac{1}{12}.P_{k,0}.\cos^2(\phi_k).T_c^3.r_{a_k,a_1}
\end{aligned}
\qquad \text{(A.20)}
$$

From (A.20), by assuming that ϕ_k is uniformly distributed over $[0,2\pi]$, we can write:

$$
\begin{aligned}
Var\left[\Phi_{0,1,a_1,k,a_k}\right] &= \frac{1}{T_b}\frac{1}{12}.P_{k,0}.\mathrm{E}\left[\cos^2(\phi_k)\right].T_c^3.r_{a_k,a_1} \\
&= \frac{1}{M^3}\frac{1}{24}.P_{k,0}.T_b^2.r_{a_k,a_1}
\end{aligned}
\qquad \text{(A.21)}
$$

Finally, the average Signal to Interference plus Noise Ratio from the first transmitter $SINR_{0,1,a_1}$ at the reference receiver 0 matched with the code sequence $a_1(t)$ is given by

$$
\begin{aligned}
SINR_{0,1,a_1} &= \frac{D^2_{0,1,a_1}}{\sum_{k=2}^{K} Var[\Phi_{0,1,a_1,k,a_k}] + Var[\eta]} \\
&= \frac{\frac{1}{4} T_b^2 . P_{1,0}}{\sum_{k=2}^{K} \frac{1}{24M^3} . P_{k,0} . T_b^2 . r_{a_k,a_1} + \frac{1}{4} N_0 T_b} \\
&= \frac{P_{1,0}}{\frac{1}{6M^3} . \sum_{k=2}^{K} . P_{k,0} . r_{a_k,a_1} + \frac{N_0}{T_b}}
\end{aligned}
\tag{A.22}
$$