Network Time Synchronization and Code-based Scheduling for Wireless Ad Hoc Networks

By

Carlos H. Rentel

A Thesis submitted to

The Faculty of Graduate Studies and Research

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Ottawa-Carleton Institute of Electrical and Computer Engineering

Department of Systems and Computer Engineering

Carleton University

Ottawa, Ontario, Canada

January 17th 2006

The undersigned recommend to the Faculty of Graduate Studies and Research

acceptance of the thesis

Network Time Synchronization and Code-based Scheduling for Wireless Ad Hoc Networks

submitted by Carlos H. Rentel

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

_____

Chair, Department of Systems and Computer Engineering
Dr. Rafik Goubran

_____

Thesis Supervisor
Dr. Thomas Kunz

_____

External Examiner
Dr. J. J García-Luna-Aceves

Carleton University

January 2006

Abstract.

This thesis addresses the study of a network time synchronization (NTS) algorithm for wireless Ad Hoc networks (WAHNs), and the design and analysis of a medium access control (MAC) paradigm referred to as code-based scheduling.

We present a novel network time-synchronization algorithm referred to as Clock-sampling Mutual Network Synchronization (CSMNS) that shows excellent scalability, accuracy and low implementation complexity. CSMNS is evaluated analytically and numerically in single-hop and multi-hop scenarios. The performance of CSMNS is also compared to the performance of the timing synchronization function (TSF) of the IEEE 802.11 standard of which, to the best of our knowledge we are the first to provide performance results in a multi-hop scenario. CSMNS has the ability to make the time difference among the clocks in the network converge to a common and small value by utilizing the timing information carried on beacons originated in any point of the network. CSMSN also reduces the need for constant refreshment of the timing information.

A MAC strategy we call code-based scheduling is proposed that utilizes the code-words of codes traditionally used for channel coding purposes. We use a novel approach that utilizes coding theory concepts to devise a scheduling strategy that allows for the possibility to guarantee a minimum level of performance in the nodes of a WAHN. General principles of code-based scheduling are investigated and particular examples based on Reed-Solomon and Hermitian codes are evaluated analytically and numerically in terms of their average, minimum and maximum delay and throughput performance.

Additionally, a metric that can be used to identify better codes for the purpose of code-based scheduling is identified.

The average performance of a large family of code-based scheduling protocols is analytically compared to the one obtained by slotted-ALOHA. A code-selection algorithm is proposed that can improve the average throughput of code-based scheduling when the number of nodes in the network is larger than the number of code-words available. Finally a hybrid code-based contention-based scheduling protocol is discussed and evaluated, which combines the performance guarantee advantage of a code-based scheduling approach with the better average performance of contention-based scheduling protocols with feedback.

Acknowledgements

I would like to thank my graduate supervisor Professor Thomas Kunz for his excellent support and dedication during my PhD studies at Carleton University. His academic rigor and experience helped me develop excellent research habits, and improve the quality of this thesis. I would also like to thank the members of my committee for their valuable comments and suggestions. Thanks to my entire family for their support and understanding. Thanks to my mother to whom I own my life and who gives me all the humanly possible love in this world. Thanks to my father for his support, unconditional love, and guiding spirit. I am thankful for having such a nice and supporting family, my brother Gustavo, and my sisters Maria, Ida, and Monica. I am also thankful to the people at the Eaton Innovation Center in Milwaukee, specially Luis Pereira, Sujit Das, Brian Armstrong, and José Gutiérrez.

I am eternally grateful to my wife Carmen to whom I dedicate this work. Your support, sacrifice and understanding are my light and my energy. Thanks to my daughters Astrid and Andrea who fill my life with joy and purpose. All these are the people worth living for. I also dedicate this work to all those who are misjudged and unfairly treated, to those who fight for their dreams and move mountains, to those that forgive and persist…

Table of Contents

List of Tables

List of Figures

List of Acronyms

AG	Algebraic Geometry
ACK	Acknowledgement
BCH	Bose-Chaudhuri-Hochquenghem code
CDMA	Code Division Multiple Access
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
CSMNS	Clock Sampling Mutual Network Synchronization
CSMNS-RMN	Clock Sampling Mutual Network Synchronization – Rotating Master Node
DE-RS	Doubly Extended Reed-Solomon code
DSSS	Direct Sequence Spread Spectrum
EANT	Ensemble Average Network Throughput
FHMA	Frequency Hopping Multiple Access
FHSS	Frequency Hopping Spread Spectrum
GF	Galois Field
Her	Hermitian code
IBSS	Independent Basic Service Set
IEEE	Institute of Electrical and Electronic Engineers
IP	Internet Protocol
MAC	Medium Access Control
MDS	Maximum Distance Separable
MEMS	Micromechanical Systems

NTS	Network Time Synchronization
OA	Orthogonal Array
OFDM	Orthogonal Frequency Division Multiplexing
OSI/ISO	Open System Interconnection / International Organization for Standardization
PHY	Physical Layer
PLL	Phase Locked Loop
RBS	Reference Broadcast Synchronization
RS	Reed-Solomon code
SANoT	Sample Average Throughput
SE-RS	Singly Extended Reed-Solomon code
TBTT	Target Beacon Transmission Time
TDMA	Time Division Multiple Access
TSF	Timing Synchronization Function
UTC	Coordinated Universal Time
VID	Vector Identifier Polynomial
VFO	Variable Frequency Oscillator
WAHN	Wireless Ad Hoc Network

Chapter 1

Introduction

In this work we study the feasibility of achieving network time synchronization (NTS) for wireless Ad Hoc networks. Additionally, we present a novel Medium Access Control (MAC) protocol for Wireless Ad Hoc networks. Novel NTS and MAC approaches are proposed and evaluated that are referred to as Clock-sampling Mutual Network Synchronization (CSMNS) and Code-based scheduling respectively. This chapter presents the fundamental concepts, motivation, objectives, and contributions of this work.

1.1 Wireless Ad Hoc Networks

A wireless Ad Hoc network is a distributed communication network comprised of geographically separated radio terminal units or nodes (mobile or fixed) that participate in the creation, management and data forwarding operations of the network over a wireless transmission medium. A wireless Ad Hoc network is autonomous and operates by the shared responsibility of the entities that provide the communication service itself; all or part of the nodes implement the required tasks to be host, router and transmission medium interface. In this respect, a wireless Ad Hoc network differs from a cellular radio network in that its functionality is not strongly dependent on the use of a centralized and fixed infrastructure. In terms of network architecture, we visualize a wireless Ad Hoc

network as a general concept that can encapsulate networks with no infrastructure or networks with some degree of infrastructure support (i.e., clustered networks). Furthermore, a wireless Ad Hoc network is not necessarily an isolated entity since it can be attached to existing network infrastructure (e.g., a multi-hop network with gateways to the Internet). The same gain in flexibility offered by a wireless Ad Hoc network introduces many degrees of freedom that can be challenge to tackle. The lack of a centralized infrastructure and the broadcasting nature of the wireless medium make it difficult to schedule transmissions, route the information through the network, connect to other existing networks, maintain efficient medium utilization, guarantee some level of quality of service (QoS), and perform network discovery among others. Furthermore, the specific applications towards which some Ad Hoc networks are tailored makes the relatively simple and general layered design of a communication system based on the OSI/ISO model a sub-optimum one, therefore cross-layer designs have been proposed that further increase the design complexity of a wireless Ad Hoc network.

Some of the most interesting advantages of Wireless Ad Hoc networks are the possibility of multi-hop mode of communication, the lack of a fixed and centralized infrastructure, and the capability of self-organization. These advantages made them initially attractive for battlefield and disaster relief scenarios in which the deployment of a fixed infrastructure can be costly, risky, and time-consuming. However, more applications have emerged recently due, in a large extent, to the field of wireless sensor networks. Wireless sensor networks are wireless Ad Hoc networks in which the nodes comprising the network have the capability to interact with the surrounding medium for distributed monitoring or control purposes. Each node is a collection of processor(s),

memory, and I/O units (e.g., radio interface, sensing/actuating elements, and mobility components) that is able to coordinate efforts with other nodes in the network in order to perform a specific task. Each node could operate in an unattended manner or with the minimum intervention of an operator. Sensor network applications are numerous, including monitoring of physical, chemical or environmental phenomena over hostile, remote and/or large geographical areas, distributed data extraction, and control and actuation (e.g., switching on/off of geographically separated devices in response to manual commands or sensed phenomena) among others. Recent advances in microelectronics and micromechanical systems (MEMS) have opened the venue for very exciting applications using small and low-cost nodes for Wireless Micro-sensor Networks [1]. In general, wireless Ad Hoc networks have been recognized as an important application enabler, not only for the military, but also for civil, industrial and scientific applications.

The first part of this thesis addresses the problem of time synchronizing the nodes in a wireless Ad Hoc network. The nodes must be able to generate, in a distributed way, a variable T that takes progressive values as time evolves. The variable T in each node of the network is called a time-process and the whole set of variables T generated independently (i.e., one per node) must show the same value within some error tolerance at any given time. That is, a procedure exists throughout the network such that for any $T_i(t)$ and $T_j(t)$ with $i \neq j$, $\left|T_i(t) - T_j(t)\right| \leq e$, where e is as close to zero as needed, and $T_i(t)$ denotes the time process of the i^{th} clock. In colloquial terms, each node has a clock that shows approximately the same time as the rest of the clocks embedded in every node of the network. The process T does not necessarily have to follow real time or be

implemented with physical clocks as will be explained in more detail in Chapter 2. Figure 1.1 depicts a view of a wireless Ad Hoc network with time-synchronized nodes and their time processes.

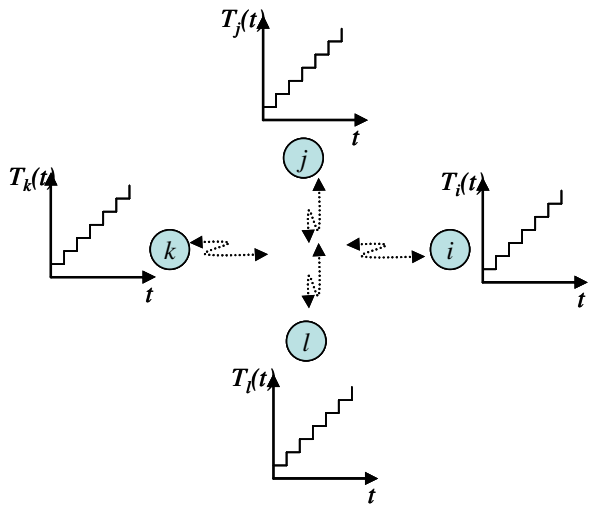

Figure 1.1 Time-processes in a Wireless Ad Hoc network in which nodes are time-synchronized

The identification of NTS as an important functionality of a network has been recognized since early times. NTS is used in almost any practical communication network. Cellular radio standards, such as the Global System for Mobile Communications (GSM) utilizes NTS to coordinate the transmissions between the mobile terminals and the base stations. The Code Division Multiple Access (CDMA) IS-95 standard utilizes NTS to maintain the orthogonality between the Walsh functions used to separate the traffic flows in the forward link, and for inter-base station synchronization among others. Within the realm of wireless metropolitan, local, and personal area networks, the IEEE 802.11 standard [2] in its "Ad Hoc" and infrastructure modes utilizes NTS for power management procedures aimed at reducing energy consumption and

increasing battery life. The IEEE 802.16 [3] as well as the IEEE 802.15 [4], [5] standards depend on an NTS function for the creation of a time-slotted medium access control protocol (i.e., a synchronous MAC).

NTS can enable many existing and yet to be discovered protocols and applications. Some of these are:

1. **A synchronous MAC protocol** that can deliver the information with some level of QoS support. In particular, NTS can allow the utilization of the wireless medium by multiple nodes in a time division multiple access (TDMA) manner. A TDMA scheme is more suitable than a contention or random-based scheduling approach to guarantee some level of delay and throughput performance in the transmission of information. It is interesting to note that many synchronous MAC protocols proposed in the research literature assume NTS is available through the Global Positioning System (GPS). However, depending on GPS signals is, in most cases, less flexible and more costly due to the need of additional hardware and clear-sky line-of-sight. MAC protocols, such as those based on combinations of any of the three traditional ones (i.e., CDMA, TDMA, and FDMA) could depend on an NTS function, as well as contention-based approaches that make use of slotted-time (e.g., slotted-ALOHA, Packet Reservation Multiple Access-PRMA [6]-[10]).
2. **Energy saving** through techniques that involve sleep/wake-up modes of operation, or through the reduction of communication overhead. It is known that most of the energy consumed by a node is due to the energy consumed by the RF section of the radio transmitter. Therefore, techniques that avoid the frequent need for the transmission/reception of information while maintaining a good performance are

highly desirable. For instance, in sensor networks it is possible to accomplish this if a node can predict the state of neighboring nodes with some gradient information and a common time measure. In particular, a node can have information of the rate of change of the phenomenon observed by its neighbor node and predict the value of the measured variable (as seen by the neighbor node) if a common time process exists between the two nodes. In other words, transmission of information can be replaced by local computation.

3. **Data aggregation** is an important paradigm in which the collection of distributed information (distributed in space and time) is ultimately presented in a concise and accessible manner to the final user. To make sense out of the distributed information it is important that a common time process exists in order to identify the proper order of events detected or monitored in the network. Data aggregation depends on NTS to avoid duplicates, and to be able to identify true updates made by the distributed nodes of the network.
4. **Security protocols** that, for instance, detect and defend against wormhole attacks through temporal leashes [11]. In a wormhole attack an attacker receives packets in one point of the network and tunnels them, via a dedicated link, to another attacker located in a different position in the network. The receiving-end attacker will re-play the packets transmitted at the other end causing a disruption of the normal operation of the network, and in particular, the routing protocol. A temporal leash uses timestamp information sent in every packet to estimate the time a packet has traveled. The receiving node performs a time comparison between the received timestamp and its own time therefore creating the need for a tight NTS function.

5. **Sequential tasks** in which some nodes respond or interact with other nodes based on a specific interval of time (e.g., interactive toys, sequential procedures in an industrial plant, wireless control).

How to achieve NTS in a wireless Ad Hoc network represents the first part of this thesis. The second part tries to answer the question on how to effectively access the wireless medium once the network is synchronized.

1.2 Network Time Synchronization

Time measurement has intrigued researchers for a long time. The first accurate clocks started to appear in the 17th century after the scientific work pioneered by Galileo and expanded by Christian Huygens [12]. These scientists helped develop a better clock based on their theories of the motion of pendulums. An important use of clocks was for determining the longitude of ships at sea, for which a sextant and an accurate clock was needed. Furthermore, people needed to agree on the time their clocks were showing, therefore means to synchronize multiple and independent clocks to a central clock were essential. This is when the Greenwich Mean Time (GMT) was born in the 19^{th} century. More recent advances in the area of clocks and time measurement have replaced GMT time (based on a physical clock located in Greenwich) with the Coordinated Universal Time (UTC) that is based on more accurate time measurements collected from different atomic clocks throughout the world. Relatively recent advances in the area of clocks include the development of the atomic clocks that are used in the GPS satellite system and the digital telephone communication network [13], the development of the quartz-

crystal oscillator, and more recently the development of a chip-scale atomic clock that could potentially bring atomic clock precision to everyday-life devices such as cell-phones and computers [14].

NTS has the goal of aligning or equalizing the individual time processes of a network of clocks separated in space. In order to achieve NTS it is necessary to establish links of communication between the clocks comprising the network. Those links could be wired or, more importantly for our study, wireless links.

There are two commonly known approaches for clock synchronization based on the network used to distribute the timing information [15], centralized and decentralized. The centralized synchronization approach is also known as master-slave and it is the most common method encountered in practice in civilian applications. There are one or more accurate clocks (the master(s)) to which all the rest of the clocks listen and adjust their frequency and phases accordingly. The decentralized synchronization approach is also known as mutual synchronization, in this approach there is no master clock, but instead all clocks cooperate to achieve synchronization in a distributed manner. In mutual synchronization, the clock of a node tries to achieve synchronization by reducing its phase or timing error with respect to a weighted average of the other clocks' phases. This method finds wide acceptance in military networks.

In practice it is necessary to exchange timing information among the different clocks in the network. There are several approaches for doing this including:

1. Burst position measurement,
2. Continuous correlation of timing signals and,
3. Clock-sampling

In the burst position method each node schedules the periodic transmission of a burst or pulse. At each receiving node the positions of the incoming bursts are compared with the position of the local burst and the difference is used to correct the local clock period according to a many-to-one mapping. This mapping usually takes the form of a weighted average of the errors. The transmission of pulses has the disadvantage of requiring a large bandwidth, and possibly a dedicated channel.

In the continuous correlation method each node continuously transmits a signal that is tracked at the receiving node. At each receiving node the sequence is compared with a replica generated by the local clock and a sliding correlation is performed in order to compute the phase offset. For instance, a clock can drive a pseudo-noise (PN) sequence generator, and this sequence can be transmitted to other nodes. As in previous cases a many-to-one mapping is needed to extract the correction term used to adjust the local clock.

In the clock-sampling method each node reads the time of its clock and transmits it to other neighboring nodes explicitly. At each receiving node the timing errors are computed as the difference between the local and neighbor nodes' clocks. The errors are used in a many-to-one mapping rule to determine the correction applied to the local clock. The main advantage of the clock-sampling technique is its relative simplicity. For example, the Timing Synchronization Function in the IEEE 802.11 standard is a clock-sampling method. In all these methods the timing information exchanged can be corrupted during the time it travels from transmitter to receiver. Some of the most detrimental factors include link delay, signal fading, signal delay spread, and collision of timing messages due to the broadcasting nature of the wireless medium.

Another important classification of NTS can be based on the method used to achieve synchronization. A virtual NTS approach does not exert explicit control over a physical clock, but rather lets the clocks in the network run freely. Synchronization is achieved by comparing timestamps and using mathematical expressions that incorporate knowledge of the clock's characteristics (e.g., maximum drift and skew of the clocks – see Chapter 2). Virtual NTS approaches have been proposed primarily for the correct estimation of the ordering of events in a computer network. More will be discussed about virtual NTS in Chapter 2. A Physical NTS approach exerts control over the physical clocks and tries to explicitly adjust their time based on a time-error measurement. Table 1.1 summarizes the advantages and disadvantages of some NTS approaches.

1.3 Medium Access Control

The efficient schedule of the transmissions in a Wireless Ad Hoc network is a challenging task. This challenge arises due to the lack of a centralized infrastructure and the broadcasting nature of the wireless medium. The most popular scheduling schemes are those derived from the ALOHA protocol, such as the Carrier Sense Multiple Access /Collision Avoidance (CSMA/CA) protocol used in the IEEE 802.11 standard. It is well known that the family of ALOHA protocols offer good performance for a small number of nodes transmitting bursty traffic over a shared channel. However, ALOHA-like protocols suffer from lack of any performance guarantee and instability. Stabilization techniques, such as the Pseudo-Bayesian algorithm [16] are not the main focus of this work, but rather we are more interested in providing some level of performance guarantee

Table 1.1 Summary of advantages and disadvantages of some NTS approaches

NTS approach	Advantages	Disadvantages
Master-slave	○ Simplicity ○ Stability	○ Hierarchical and centralized with single point of failure ○ Not suitable for distributed networks and applications (i.e., becomes the weakest link) ○ Accuracy decreases as slaves are farther away from the master (e.g., multi-hop or multiple master hierarchies)
Mutual Synchronization	○ Non-hierarchical architecture (all nodes have the same influence), suitable for distributed networks ○ Accuracy does not degrade on a multi-hop network	○ Complexity ○ Stability issues due to close-loops ○ Accuracy depends on link delays when not negligible
Virtual	○ Simplicity in some cases ○ Potential trade-off of transmission overhead for local computation	○ Depends on knowledge of clock parameters and assumptions ○ Loose accuracies
Physical	○ Accuracy	○ Complexity of implementation.

through the scheduling procedure. It is necessary to have more structure in the scheduling approach if real-time or priority applications are to be supported with some minimum performance guarantees. A way to add structure is by allowing the transmissions to happen during known boundaries or slots. This technique is employed in the slotted-ALOHA protocol. However, a high degree of randomness is still present in the way nodes access the medium, therefore no delay guarantees can be offered. On the other hand, a highly structured scheduling protocol usually suffers from being inflexible and difficult to manage in a distributed way.

An example of a structured scheduling protocol is classic time division multiple access (TDMA). In classic TDMA every node is assigned a unique time-slot in which to transmit; in this way collisions are completely avoided and the packet delay becomes highly deterministic. However, the throughput and delay efficiency of such a scheme is questionable due to the poor re-utilization of slots (i.e., nodes far enough from each other may utilize the same slot to transmit without causing interference). Other more intelligent TDMA schemes have been proposed that try to be more efficient. There are several requirements that can be imposed in order to design an efficient method for multiple nodes to access the transmission medium, some of the most important in the context of wireless Ad Hoc networks include: 1) Minimization of primary and secondary collisions, 2) Maximization of slot re-utilization, and 3) Minimization of overhead. A primary collision occurs when two nodes within interference range of each other transmit in the same time-slot; in a primary collision the set of intended receivers of the two transmissions necessarily include one or both of the transmitting nodes. This is assuming a node cannot transmit and receive at the same time.

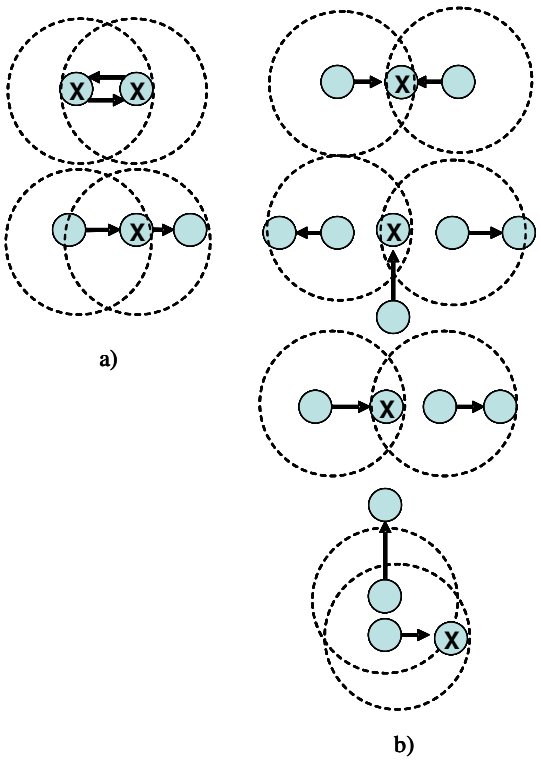

Figure 1.2 Examples of primary (a) and secondary (b) collisions

A secondary collision occurs when two nodes, not necessarily within interference range of each other, transmit in the same time-slot, interfering with an ongoing reception in a third node that is located within interference range of both transmitting nodes; in a secondary collision the node in which the collision happened may or may not be the intended receiver of the two transmitters.

Figure 1.2 depicts some scenarios in which primary (2.a) and secondary (2.b) collisions occur. The dotted circles surrounding the nodes represent the transmission range of a node, and the x inside a node identifies the location of the collision. A collision is usually assumed to destroy the information at the given receiver. However, in practice,

the receiver might be able to capture the information if the information is received with sufficient power above a given noise and interference threshold. In this work we assume no capture. Ideally, in order to avoid collisions entirely, any two transmitting nodes must use different time-slots if at least one of the two intended receivers is within interference range of both transmitting nodes in question. Conversely, if the intended receivers of the two transmitting nodes are out of the interference range of the corresponding non-intended (interfering) transmitting node, then the two transmitting nodes can transmit in the same slot. For instance, in the primary collisions of Figure 1.2a, all transmitting nodes must use different time-slots because the intended receivers are either one or both of the transmitting nodes themselves, which in turn are within transmission range of each other. In the secondary collisions of Figure 1.2b, the example on top and the last two involve at least one intended receiver that is within interference range of the two transmitting nodes, therefore all transmitting nodes must use different time-slots. However, in the second example from top to bottom, the two nodes transmitting in the horizontal direction can transmit in the same time-slot because their intended receivers are out of interference range from the corresponding interfering transmitter (i.e., for simplicity assume that the transmission and interference ranges are equal to the radius of the circles shown). Note from the previous discussion that slots can be re-used even by nodes within transmission range of one another. However, the amount of overhead information needed to achieve slot-reutilization could be substantial if it is explicitly sought.

The maximization of slot-reutilization is an important factor since it translates into the minimization of the frame size (i.e., number of slots per frame), which in turn translates into reduced transmission delays and increased bandwidth availability for a given node.

Consider for instance the network of Figure 1.3, in which nodes that are not linked by a line do not interfere with one another. A classic-TDMA scheme will assign a different slot to every node, and the resulting frame size will be of 6 slots. However, it is possible to achieve the same collision-free scenario of classic-TDMA by using a frame with 3 slots, reducing to half the time it takes a node to successfully remove a packet from its transmitting queue. However, it is clear that achieving slot-reutilization implies some way of distributing the information of what slots a given node will use. The latter is not a trivial task particularly if the topology of the network changes frequently due to node mobility.

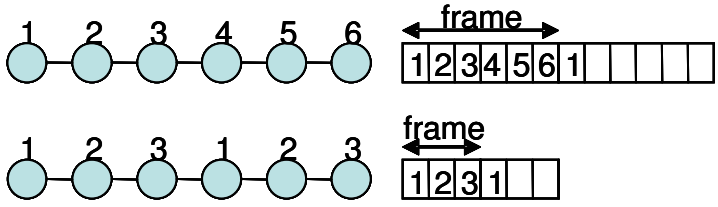

Figure 1.3 Example of slot-reutilization

In order to assign slots in a WAHN we could adopt one of two approaches: 1) a topology dependent or 2) a topology transparent approach. A topology dependent approach will gather and distribute topological information necessary to realize successful broadcasting and/or unicasting transmissions. A topology dependent scheduling approach will adapt to any change in topology, either in a predictive or reactive way to attain the desired scheduling objective. We argue that a topology dependent approach will necessarily incur excessive overhead due to the lack of an accurate approach to predict the state of a highly dynamic wireless Ad Hoc network. However, a topology dependent approach could be a good candidate in situations in

which the network dynamics are slow. A topology transparent approach, on the other hand, will try to schedule the slots with no topological information at all. However, it is generally accepted that a topology transparent approach can make use of global network information such as the number or density of nodes in the network. A simple topology transparent approach would, for instance, assign time-slots randomly to all nodes wishing to transmit in a given time-slot (i.e., slotted-ALOHA). The MAC approach proposed in this work is topology-transparent.

1.4 Research Objectives

The main objective of this work is two investigate and propose new network time synchronization and MAC mechanisms for wireless Ad Hoc networks in order to provide quality of service guarantees to time-constrained applications. The specific goals are:

1. To design a WAHN Time Synchronization algorithm with the following characteristics:
    a. **Accurate**: Enough accuracy to allow for a time-slotted MAC protocol (i.e., few μsec accuracy)
    b. **Scalable**: Support of hundreds of nodes is desirable for future WAHNs.
    c. **Non-hierarchical and autonomous**: Every node must have the same influence over the NTS performance, and the information needed to achieve NTS should be derived from the network itself,

d. **Convergent**: Different to other NTS approaches (e.g., the TSF of IEEE 802.11, and classical mutual network synchronization), the error among the time-processes in the network should converge or approach zero as time evolves.

e. **Distributed**: The NTS approach should be fault tolerant without central points of failure.

f. **Low overhead**: The achievement of NTS should not be costly in terms of overhead. An NTS approach should not affect substantially the performance of the network in terms of its main task (i.e., data communication)

g. **Simple**: The approach needs to be simple enough to allow for its implementation using most of the existing lower-layer technology, such as IEEE 802.11 or IEEE 802.15.4. While mutual synchronization is deemed a complex and costly approach, the reality is that to the best of our knowledge there is no study that tries to reduce this complexity even if it means degradation in performance.

2. To design a WAHN MAC strategy that is:

a. **Resilient to topology changes**: A topology-transparent scheduling approach is preferred due to its lower overhead potential,

b. **Guarantee performance** for priority (e.g., control/signaling channel) or real-time applications,

c. **Fair**: The use of the wireless medium for the transmission of information should be fair among equivalent nodes.

d. **Efficient**: A good reutilization of slots to ensure a good average delay and throughput performance.

1.5 Research Contributions

The main contributions of this work are:

1. The discovery of an NTS algorithm referred to as CSMNS that satisfies all the requirements specified in Section 1.4 and the introduction of the important concept of convergence in the context of NTS algorithms for wireless Ad Hoc networks. To the best of our knowledge, we were also the first to present performance results of the IEEE 802.11 TSF in a multi-hop scenario,
2. A generalized view of the theory of topology-transparent scheduling protocols [17] through the incorporation of coding theory concepts into the MAC problem. We are the first to compare the use of Reed-Solomon and Hermitian codes in the context of scheduling of transmissions for wireless Ad Hoc networks. Furthermore, we were the first to explicitly identify the relative minimum distance of a code as a parameter that can be used to compare the performance of different codes to access the medium with some minimum level of success. We analytically compare a contention-based scheduling protocol (i.e., slotted-ALOHA) with a code-based scheduling protocol along with the description and evaluation of an algorithm used to improve the average throughput performance of code-based scheduling protocols referred to as the code-selection algorithm. Additionally, we introduced a hybrid code-contention based scheduling approach that combines the benefits of a contention-based scheduling protocol and a code-based scheduling protocol.

The following is a list of publications and presentations accepted so far:

1. Carlos H. Rentel and Thomas Kunz, "MAC coding for QoS guarantees in multi-hop mobile wireless networks," Proceedings of the 1st ACM International Workshop on QoS and Security for Wireless and Mobile Networks, Montreal, Canada, October 2005, p. 39-46.
2. Carlos H. Rentel and Thomas Kunz, "Reed-Solomon and Hermitian code-based scheduling protocols for wireless ad hoc networks," Proceedings of the 4th International Conference on Ad-Hoc Networks and Wireless (Springer Lecture Notes in Computer Science 3738), Cancun, Mexico, October 2005, ISBN 3-540-29132-6, p. 221-234.
3. Carlos H. Rentel and Thomas Kunz, "On the average-throughput performance of code-based scheduling protocols for wireless ad hoc networks," ACM Poster, MobiHoc 2005, Urbana-Champaign, USA, May 2005.
4. Carlos H. Rentel and Thomas Kunz, "A clock-sampling mutual network synchronization algorithm for wireless ad hoc networks," Proceedings of the IEEE Wireless Communications and Networking Conference, New Orleans, USA, March 2005, p. 638 – 644.
5. Carlos H. Rentel and Thomas Kunz, "A non-hierarchical and convergent timing synchronization function for real-time QoS support in wireless LANs: Towards an autonomous network synchronization algorithm for wireless ad hoc networks," Proceedings of the Wireless World Research Forum Meeting 12, Toronto, Canada, November 2004.

6. Carlos H. Rentel and Thomas Kunz, “A distributed network synchronization algorithm for wireless ad hoc networks,” ACM Poster, MobiCom 2004, Philadelphia, USA, October 2004.

The outline of this thesis is as follows: Chapter 2 discusses NTS, mathematical models and previous work in the area. Chapter 3 describes, analyses, and evaluates the proposed NTS algorithm (CSMNS). Chapter 4 discusses topology-transparent scheduling protocols. The proposed code-based scheduling is presented in Chapter 5. Finally, Chapter 6 presents the final conclusion and discussion of this work along with some guidelines for future work.

Chapter 2

Network Time Synchronization

This chapter addresses the topic of Network Time Synchronization (NTS) in the context of Wireless Ad Hoc Networks. We start with the definition and mathematical model of a discrete NTS approach. We continue with the description and discussion of some important related work on NTS, followed by a description and analysis of the Timing Synchronization Function (TSF) of the IEEE 802.11 standard.

2.1 Mathematical Model of Network Time Synchronization

The NTS method proposed in this work makes use of physical clocks to define the time-process generated in every node of the network. The use of physical clocks does not imply interest in obtaining a precise measure of time in terms of Coordinated Universal Time (UTC). It is sufficient for our purposes to obtain time synchronization within some error accuracy regardless of the absolute value shown by the time-processes. Furthermore, the proposed NTS approach is more suitable for a multi-hop wireless network either mobile or static or a single-hop infrastructure-less network. We start by defining and characterizing the fundamental building block of any NTS approach.

2.1.1 Mathematical Model of a Clock

A clock is a time measurement device. It is a system comprised of an oscillator and an accumulator or counter. The oscillator's task is to generate periodic events and the counter adds-up these events in order to obtain the measured time in the form of timestamps. For instance, the oscillator can produce a sinusoidal waveform at its output; and the zero-crossing events of this sinusoidal waveform can be detected and used by the counter to increment the value of its output, which ultimately has the form of a timestamp. The absolute value shown by the counter output is unimportant as long as it follows a consistent rule throughout the entire network of interest. The synchronization with UTC necessarily implies the need for connecting to an external network, such as GPS or the Internet.

It is clear that the core functionality of a physical clock is its oscillator. The performance of the oscillator in generating accurate periodic events will ultimately determine how accurate the clock is. Therefore, it is traditional to start modeling the clock by looking at the output of an oscillator. The output of an oscillator can be modeled with the mathematical expression of a real periodic waveform. A function $f(t)$ is a real-periodic waveform with nominal period T_o if

$$f(t) = f(t + T_o + e_p(t)).$$

Where $e_p(t)$ is an error due to the imperfections of the oscillator, and it is, in general, a function of time. The specific amplitude values of the oscillator are unimportant for our

model. However, it is clear that the possible functions $f(t)$ do not include the trivial case of a constant function in time (i.e., $f(t) \neq k$, where k is a constant value). The clock imperfections are due to ambient conditions such as temperature, relative humidity, pressure, and imperfections in the methods and materials used in the construction of the oscillator. The errors in the generation of a true-periodic waveform are characterized by the phase differences between the waveform generated by the oscillator under test, and a reference-oscillator. If the reference-oscillator is taken to be a perfect oscillator with frequency $w(t) = w_0$ (and phase $\theta(t) = w_0 t$), then the phase difference is usually modeled as [18, 19]

$$\Phi_e(t) = \Phi_{e0} + w_{e0}t + \frac{1}{2}Dt^2 + \xi(t), \tag{2.1}$$

The phase error is measured using a phase detector that subtracts the phases of the oscillating waveform under test and the waveform produced by the reference-oscillator. $\Phi_e(t)$ is the phase error as a function of real time, Φ_{e0} is the initial phase error, w_{e0} is the fixed frequency offset of the oscillator with respect to the nominal frequency w_0, D is the frequency drift coefficient, and $\xi(t)$ is a random process that models the short-term phase variations of the oscillator. In synchronization parlance the D coefficient contributes to the *frequency wander* of the oscillating waveform with respect to a reference, and the $\xi(t)$ term contributes to its *jitter*. The $\xi(t)$ term is a random process that has been characterized with a power-law spectra [19]. In other words, the Fourier transform of the phase errors between an oscillator and its reference has been found to be of the form f^b,

where b identifies the power of the process. If $b = 0$, the process is a white noise process. All accurate clocks[♣] exhibit this random behavior. However, for most applications, these jittering disturbances can be neglected.

Equation (2.1) shows that the phase difference between the perfect reference-oscillators and the test-oscillator have a systematic component and a random component ($\xi(t)$). The systematic component is comprised of a constant, a linear, and a quadratic increasing set of functions.

An ideal oscillator will have a zero frequency drift (i.e., $D = 0$ Hz/day), and no short-term variations ($\xi(t) = 0$), assuming the initial phase (Φ_{e0}) and frequency offset (w_{e0}) errors can be corrected. The frequency error between the test and reference oscillators can be found by differentiating (2.1), and is given by

$$w_e(t) = w_{e0} + Dt + \dot{\xi}(t) \tag{2.2}$$

As time progresses the frequency of the test-oscillator drifts linearly from the frequency of the reference oscillator. It is possible however to estimate the drift coefficient and adjust the phase of the oscillator accordingly, but no estimation is perfect and therefore drifting of the frequency will eventually dominate the frequency error of the clock. Equation (2.1) is still an imperfect model since the oscillator's frequency and

---

[♣] In this work an accurate oscillator is considered to be one based on quartz crystals up to atomic standards with frequency accuracies in the order of tens of ppm (parts per million). The IEEE 802.11 and the IEEE 802.15.4 standards, for instance, specify a clock-accuracy for their Pseudo-Noise (PN) sequence generator not greater than ±25ppm and ±40ppm respectively [2], [5]. Clocks in personal computers utilize quartz-crystal oscillators with frequency accuracies in the range of 10 to 100ppm.

phase also depend on ambient variables such as temperature, humidity, and pressure. For these reasons, it is common practice to adjust the clock timing process utilizing a combination of several methods such as drift coefficient estimation, temperature compensation, and time refresh coming from one or several reference-clocks. Using (2.2), the frequency of the oscillator is

$$w(t) = w_0 + w_{e0} + Dt + \dot{\xi}(t) . \tag{2.3}$$

If (2.3) is integrated in time we obtain the phase of the oscillator

$$\Phi(t) = \int_0^t \left(w_0 + w_{e0} + Dt + \dot{\xi}(t)\right) dt = w_0 t + w_{e0} t + \frac{D}{2} t^2 + \xi(t) - \xi(0) + \Phi(0) . \tag{2.4}$$

Where $\Phi(0)$ is the initial phase and the rest of the parameters were defined before. Ideally, a perfect clock's phase would be

$$\Phi_{ideal}(t) = w_0 t . \tag{2.5}$$

Therefore dividing the phase in (2.4) by the nominal frequency of the oscillator we obtain the measured time of the clock modeled according to (2.1)

$$T(t) = t + \frac{w_{eo}}{w_o} t + \frac{D}{2w_O} t^2 + \frac{\xi(t)}{w_O} + \Theta(0), \tag{2.6}$$

where $\Theta(0) = \Phi(0) - \xi(0)$.

The measured time *T(t)* is a random process with systematic and random components. Several important terms that characterize the performance of a clock are defined based on the time process *T(t)*.

The *offset* of a clock *a* with respect to a clock *b* (or reference time *t*) is $T_a(t) - T_b(t)$ (or $T_a(t) - t$). The *skew* is the difference between the rate of change of time of two clocks, $\dot{T}_a(t) - \dot{T}_b(t)$ (or $\dot{T}_a(t) - 1$), where $\dot{T}(t) = {dT(t)}/{dt}$, and the *drift* is defined as $\ddot{T}_a(t) - \ddot{T}_b(t)$ (or $\ddot{T}_a(t)$). Taking (2.6), and assuming that $\xi(t) = 0$ and $\Theta(0) = 0$, we obtain the following for two clocks *a* and *b* with the same nominal frequency

$$\text{Offset} = \left(\frac{w_{ae0} - w_{be0}}{w_0} \right) \cdot t + \left(\frac{D_a - D_b}{2w_0} \right) \cdot t^2, \tag{2.7}$$

$$\text{Skew} = \left(\frac{w_{ae0} - w_{be0}}{w_0} \right) + \left(\frac{D_a - D_b}{w_0} \right) \cdot t, \tag{2.8}$$

$$\text{Drift} = \left(\frac{D_a - D_b}{w_0} \right). \tag{2.9}$$

The offset in Equation (2.7) indicates the difference, in units of time, of the two time processes as real time evolves, skew (2.8) can be interpreted as the rate of change of offset in units of time over time, and drift (2.9) is the rate of change of skew in units of frequency over time. A more general equation modeling a clock will include multiple derivatives of the offset beyond the second derivative (see, for instance, [15]), but this accurate analysis is beyond our interest. We assume the drift D and random component $\xi(t)$ are equal to zero in our simulations and analysis. Therefore each time process can be represented as a straight line with a specific slope and initial value (i.e., only initial setting error and skew is considered non-negligible in our model). The latter is a reasonable assumption since we are not dealing with the precise measurement of time, but rather the relative time synchronization of clocks in a network. The most common scenario for a crystal oscillator used in telecommunication radios is to be at an approximately constant frequency offset from its nominal frequency (e.g., accuracy in the range 10-100 ppm [12]), therefore, the time process will be approximately a linear function of reference time.

Clocks are ultimately discrete machines. Therefore the model of the discrete time-process with respect to real-time t is given by

$$T(nT_p) = \beta \cdot t(nT_p) + T(0). \tag{2.10}$$

Where T_p is the sampling period in real-time, $t(nT_p) = nT_p$, and β is the skew of the clock with respect to real-time. Equation (2.10) defines a discrete straight line that can be represented as in Figure 1.1.

Two perfectly synchronized clocks have an offset equal to zero. In practice however, this is difficult to achieve because, among other things, the drift and the skew of the clocks are difficult to estimate. If the skew and the drift of a time-process given by (2.6) could be estimated then it would be a matter of compensating for those values in the final timestamp reported by the clock.

A fundamental implementation principle used to achieve synchronization is that of the creation of links that can be used to exchange timing information among the clocks. Each clock can use the timing information from a reference clock, or from any other clock to adjust its time. The presence of links connecting clocks creates what is referred to as a network of clocks.

2.1.2 Mathematical Model of a Discrete Network Time Synchronization Algorithm

In a network of clocks, each clock that is part of a NTS approach follows a pre-determined rule to adjust its time process according to information coming from one or more clocks in the network. Assume a network of *N* clocks connected with links. The topology of the network is unknown in general. The i^{th} clock in the network will have a time process T_i that can be expressed in a general discrete form as

$$T_i(n^+ + \max_{k \neq i}\{\Delta_k(n) + \tau_{ik}\}) = g(\mathbf{U}_i(n) \cdot \mathbf{T}(n), \mathbf{U}_i(n) \cdot \mathbf{T}_i(n))$$

$$\forall i \in [0, 1, \ldots, N-1] \text{ and } n = [0, 1, \ldots). \tag{2.11}$$

For notation simplicity we have normalized n (the adjust-time) in (2.11). However, note that the time-process progresses with the ticks of the i^{th} clock between two consecutive adjusting points. That is, the normalized sampling time n corresponds to the intervals of time at which a time-adjustment is made and not to the actual ticks of the clock, which could be smaller. $g(\,)$ is the control law that implements the adjusting rule through the information embedded in the local and received timestamps, $\mathbf{T}(n)$ is the N-1x1 column-vector of timestamps from all the clocks in the network (except the one at the i^{th} clock) at the time they are received at the i^{th} clock. $\mathbf{T}_i(n)$ is the N-1x1 column-vector of local timestamps at the specific instants in time at which the other clock timestamps are received according to the i^{th} clock. $\mathbf{U}_i(n)$ is a diagonal N-1xN-1 matrix with elements that are either one or zero and that specifies what timestamps are actually received by the i^{th} clock. The function $g(\,)$ maps an N-1 dimensional space into a real value (i.e., a many-to-one mapping) that represents the new time-stamp of the i^{th} clock immediately after the last timestamp is received (i.e., $n^+ + \max_{k \neq i}\{\Delta_k(n) + \tau_{ik}\}$). Note that Equation (2.11) characterizes those NTS approaches that exchange timing information in discrete steps. That is, the clock(s) exchange their timestamp at specific periodic intervals rather than in a continuous manner. The latter is the case in the approach we propose, and also in many practical NTS approaches (e.g., the IEEE 802.11 TSF). The start of every timing-exchange interval signals the start of a window of time during which the clock(s)

will transmit their timestamp, this is what $\Delta_k(n)$ for any $k \in [0,1,\ldots,N-1]$ in (2.11) models. The link delay between the i^{th} and k^{th} clock is denoted as τ_{ki}. The vector $\mathbf{T}(n)$ is

$$\mathbf{T}(n) =$$

$$[T_0(n)+\Delta_0(n)-\tau_{0i},\ldots,T_{i-1}(n)+\Delta_{i-1}(n)-\tau_{i-1i},T_{i+1}(n)+\Delta_{i+1}(n)-\tau_{i+1i},\ldots,T_{N-1}(n)+\Delta_{N-1}(n)-\tau_{N-1i}]^{tr}.$$

The vector $\mathbf{T}_i(n)$ is

$$\mathbf{T}_i(n)=[T_i(n+\Delta_0(n)-\tau_{0i}),\ \ldots,T_i(n+\Delta_{i-1}(n)-\tau_{i-1i}),T_i(n+\Delta_{i+1}(n)-\tau_{i+1i}),\ldots,T_i(n+\Delta_{N-1}(n)-\tau_{N-1i})]^{tr}$$

The matrix $\mathbf{U}_i(n)$ can take different forms depending on the procedure used to exchange the timing information, the network topology, and the link impairments (e.g., noise, interference, fading, physical break of the link etc.). For instance, assuming a master-slave approach with a star-like topology and highly reliable links as shown in Figure 2.1a, the $\mathbf{U}_i(n)$ matrix becomes independent of adjust-time

$$\begin{aligned}&\mathbf{U}_i = diag\{0_0,\ldots,0_{i-1},0_{i+1},\ldots 0_{j-1},1_j,0_{j+1},\ldots,0_{N-1}\},\ \forall i \in \{0,1,\ldots,j-1,j+1,\ldots N-1\}\\&\mathbf{U}_j = \mathbf{0}_{N-1xN-1}\end{aligned}.$$

(2.12)

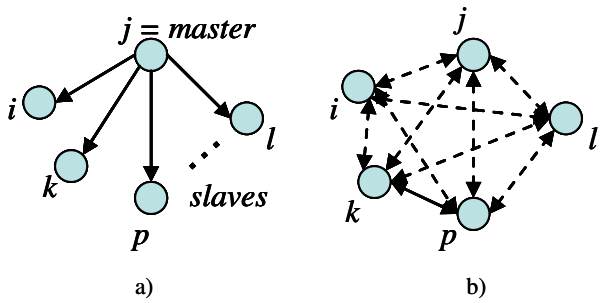

Figure 2.1 NTS approaches with different time-exchange network topologies

Where the sub-indexes used in every element of $\mathbf{U}_i(n)$ in (2.12) identify the clock that sent the timestamp (the j^{th} node is assumed to be the master). A more interesting example is a mutual NTS approach in which every node is connected to every other node using non-reliable links (e.g., wireless links), see Figure 2.1b. In this case, $\mathbf{U}_i(n)$ is

$$\mathbf{U}_i(n) = diag\{\mu_0, \ldots, \mu_{i-1}, \mu_{i+1}, \ldots, \mu_{N-1}\}, \forall i \in \{0,1,\ldots,N-1\}. \tag{2.13}$$

Where $\mu_k(n)$, $k \in [0,1,\ldots,N-1]$ is a random process that takes a discrete value of either one or zero depending on the presence of a link. The value taken by $\mu_k(n)$ depends on the impairments of the link connecting the k^{th} and the i^{th} clock ($k \neq i$), on the detection and decoding strategy used to recover from errors in the link, and on the medium access procedure used to send the timestamps.

In Chapter 3 we use a $\mathbf{U}_i(n)$ similar to (2.13) to solve the system of Equations (2.11) for CSMNS. The matrix $\mathbf{U}_i(n)$ can, in general, change at every adjusting step and have random components due to factors explained above.

The $g()$ control law plays a key role in determining the efficiency of the NTS approach. Ideally, the error vector $E_i(n) = \left|\mathbf{T}(n) - \mathbf{T}_i(n)\right|, \forall i \in [0,1,\ldots,N-1]$ should decrease to zero as *n* increases. That is $\underset{n\to\infty}{E_i(n)} \to 0$. However, this objective can be relaxed as $\underset{n\to\infty}{\left|E_i(n) - \delta\right|} \le \xi$ for given values of δ and ξ. An NTS approach that achieves the former is referred to as a convergent NTS approach and one that achieves the latter is called a marginally convergent NTS approach. A possible form of $g()$ is given by

$$g(\mathbf{U}_i(n)\cdot\mathbf{T}(n), \mathbf{U}_i(n)\cdot\mathbf{T}_i(n)) = \mathbf{K}(n)\cdot\mathbf{U}_i(n)\cdot\mathbf{T}_i(n) + \mathbf{G}(n)\cdot\mathbf{U}_i(n)\cdot(\mathbf{T}(n) - \mathbf{T}_i(n)). \quad (2.14)$$

Where $\mathbf{K}(n)$ and $\mathbf{G}(n)$ are the control *N*-1x*N*-1 matrices. CSMNS utilizes a control law with the form of (2.14). The mathematical analysis of a system comprised of equations (2.11), (2.13), and (2.14) is complicated. However, solving these set of equations numerically can provide a good insight into the behavior of the NTS approach as will be shown in Chapter 3.

2.2 Network Time Synchronization Protocols

In this section we start by presenting seminal work on NTS in the context of distributed systems, next, we will describe in some detail mutual network synchronization in a continuous-time domain. Towards the end of this section we will briefly describe some related NTS approaches recently proposed for Wireless Ad Hoc Networks, and in particular, Hu and Servetto's NTS protocol [20], and the Reference Broadcast Synchronization (RBS) approach [21].

Some NTS approaches use the client-server paradigm. A client node requests the time from a time-server periodically, or at any point in time it is deemed necessary to adjust its clock. The time-server returns the time of its clock and the local node corrects its clock based on the difference between the received timestamp and its local time. The delay of the link joining the server and the client must be compensated for if it is not negligible, additionally, the time between obtaining the timestamp to the time of the timestamp transmission at the server's side might be an important source of inaccuracies. The client-server NTS approach is depicted in Figure 2.2. The client requests the time from the time-server at local time T_{c_s}. The time-server receives the request at T_{s_r}. At time $T_{s_s}^{'}$ the time-server obtains the time from its clock and transmits it at time $T_{s_s}^{''}$. Note that the time-server may or may not be synchronized to UTC. However, this is usually the case, therefore the server time-process $T_s(t)$ is approximately equal to "real time" t (i.e., $T_s(t) \approx t$). Once the server-clock is sampled, it is possible to take two approaches regarding the value of the timestamp sent ($T_{S_{sent}}$). In one approach, $T_{S_{sent}} = T_{S_{sampled}}$, in

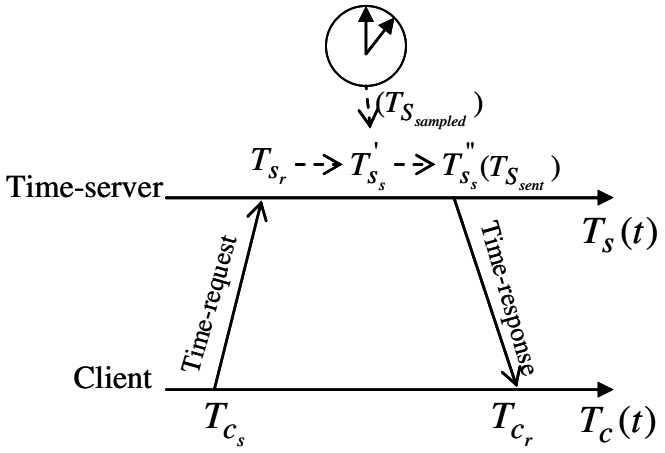

Figure 2.2 Client-server NTS approach.

which case the timestamp carries a value that shows an error of $e = T''_{s_s} - T'_{s_s}$ with respect to the actual time shown in the server-clock (assuming the reading of the clock and the creation of the timestamp packet takes negligible time). That is, if the time shown in the server-clock is $T_s(T''_{s_s})$ at the time of transmission, then $T_{S_{sent}} = T_s(T''_{s_s}) - (T''_{s_s} - T'_{s_s})$. In the second approach, the server makes an attempt to estimate the error e, and transmits a timestamp with the value set to $T_{S_{sent}} = T_{S_{sampled}} + \hat{e}$. The latter estimation might be difficult to perform, particularly if the server contends against other nodes for the transmission of the timestamp information (a packet with the timestamp value). However, as we will see later in this chapter, the combination of a mutual network synchronization approach with part of the procedure implemented in the TSF of the IEEE 802.11 standard make the accurate estimation of e a possibility. Other factors that need to be taken into account for the estimation of e are the time it takes to prepare a packet for transmission. However, this can be highly deterministic and in most cases negligible.

The client receives the timestamp at time T_{c_r} and adjusts its clock based on the timestamp received and local timestamp. The link delay must be compensated for if it is deemed considerable for the application or tasks running in the network. A simple approach is to take link delay as $\tau_l = \frac{T_{c_r} - T_{c_s}}{2}$ and either find the average of many request-response attempts, or use the one with minimum value. Note that the accuracy of this procedure depends on how fast the server is able to respond. In other words, the accuracy of the link delay estimation increases as the processing time $T_{S_p} = T_{s_s}^{''} - T_{s_r}$ decreases. Finally, the client will set its clock to $T_{S_{sent}} + \tau_l$. In case enough processing power exists in the server and the processing time T_{S_p} can be estimated, it is possible to transmit T_{S_p} (or its individual components: $T_{s_s}^{''}$, and T_{s_r}) along with $T_{S\,sent}$ to the client node. The latter can be used to estimate the link delay as

$$\tau_l = \frac{T_{c_r} - T_{c_s} - T_{S_p}}{2}. \tag{2.15}$$

The client-method approach is utilized in the Network Time Protocol (NTP) used in the Internet. NTP is a hierarchical master-slave NTS approach. The first server is synchronized to UTC, and the rest of the time-servers are synchronized in a hierarchical-tree structure. Link delay estimation is similar to the one modeled in (2.15) between servers.

A different point of view in the area of network synchronization was introduced by Lamport [22]. The fundamental ideas of Lamport are used in the TSF (see Section 2.3),

hence their importance for our work. Network synchronization is viewed as a means to find the relative order between events in processes that are separated in space. The concept of what event happened before another is studied. The distributed system is divided into separate processes that execute some tasks; each individual process is comprised of a set of events that are generated in sequence. An event can be, for instance, the execution of an instruction or a sub-program, an interruption, or the sending and receiving of a message. Figure 2.3 shows a possible depiction of this distributed system as presented in [22].

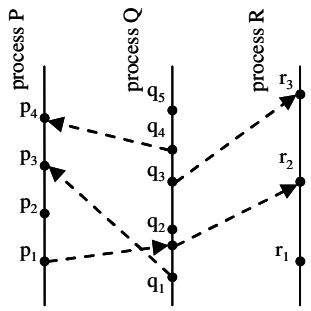

Figure 2.3 A distributed system of processes

Each vertical line represents a process with events in it depicted as dots. The message transfer between processes is denoted with dotted lines. Later times are higher than earlier ones. The important point is to have a concise and correct view of the relative order of the events throughout the system. Logical clocks that do not have an actual timing mechanism can replace physical clocks.

An event a is said to happen before an event b, denoted as $a \rightarrow b$, if some of the following relations are satisfied: 1) If a and b belong to the same process, and a comes

before b, 2) If a is the sending of a message and b is the receipt of the same message, 3) If $a \rightarrow b$ and $b \rightarrow c$ then $a \rightarrow c$. Two distinct events are concurrent if a $\not\rightarrow b$ and $b \not\rightarrow a$. An event "happens before" another one if it is possible to move forward (in upstream direction) from one to the other along the processes and the message lines in a diagram like the one shown in Figure 2.3. If two events are not connected this way they are said to be concurrent; concurrency implies also that the two events in question cannot affect one another. Note that the latter does not necessarily imply that the two events happen at the same point in time. In Figure 2.3, for instance, p_2 and q_2 are said to be concurrent even though they do not happen at the same time. However, from a practical standpoint what matters is that neither one of those two events can affect one another (regardless of when they exactly happened) since there is no connection that can take us forward from p_2 to q_2 or vice versa. In brief, two events are called concurrent if they may occur in any order, or simultaneously, without changing the outcome.

The concept of the relative order of events suggests the possibility of achieving consistency in the way an observer, located in one process, sees the order of events in the entire system without the use of physical clocks. Consequently, Lamport defines a logical clock C_i for each process P_i to be a function that assigns a number $C_i(a)$ to an event $a \in P_i$. The actual implementation of the function C_i can be done with counters rather than physical clocks.

The correctness of such a system of clocks is achieved if for any event a and b: $a \rightarrow b$ implies $C(a) < C(b)$. More specifically, two conditions are given: C.1) If a and b belong to the same process, then $C_i(a) < C_i(b)$, and C.2) If a is the sending of a message at P_i, and b is the receipt of the same message at P_j then $C_i(a) < C_j(b)$. A system is

synchronized if it satisfies C.1 and C.2 according to this view. In order to satisfy C.1 it is only necessary that for each process P_i, C_i must be incremented between any two successive events. In order to satisfy C.2 the message sent will carry the timestamp $T_s = C_i(a)$, and upon receiving a message, the process P_j sets C_j to a value greater than or equal to its present value and greater than T_s. A similar procedure is performed by the TSF in which the local time is adjusted only if the received timestamp has a later value than the local time at the moment the timestamp is received. Another procedure related to the idea of ordering the events in a distributed system, but adapted to Wireless Ad Hoc networks, is presented in [23]. The protocol for synchronization in [23] is concerned with the relative ordering of events. However, it is a reactive protocol in which the timing information is extracted only when needed. If a node *a* sends a timestamp to indicate the detection of a particular phenomenon (an event) to node *b*, then node *b* will transform the timestamp received to its local time. In this work we are not interested in finding procedures that do not use physical clocks to order events. The use of physical clocks implies the need to correct the deviations studied in Section 2.1.1. However, one of our main goals is to be able to construct a slotted time for the transmission of information, therefore the NTS approach must have a fine granularity and regularity rather than being event-driven or dependent on certain irregular or asynchronous events to trigger the need for a "time" value. The study of synchronization from the event-driven point of view is important, however, because it is necessary for data aggregation in wireless sensor networks among other things.

The algorithm proposed in this work is a mutual NTS approach. One of the first published works in this area was by Gersho and Karafin [24] who analyzed a system that

synchronized the frequencies and phases of a set of geographically separated oscillators connected by communication links. Mutual NTS is an approach that uses the internal timing information of the network to achieve network synchronization. Equation (2.11) along with the matrix in (2.13) and the depiction in Figure 2.1b show a way of representing mutual NTS in a discrete-time domain. Stability of a mutual network synchronization scheme is proved in [24] through a mathematical analysis based on classical control theory in a continuous-time domain. Geographically separated oscillators are directly controlled in a distributed manner through a multiple-input phase-locked-loop (PLL) approach. Each input of the multiple-input PLL located in every oscillator receives the timing information exchanged with neighbor oscillators utilizing the available links. Figure 2.4 shows the *phase averaging* system located in every node of the system analyzed. The transmission delay of a link joining the i^{th} and k^{th} nodes is denoted as τ_{ik}. The free-running frequency of the VFO is f_i; the control signal $r_i(t)$ affects the actual frequency of the oscillator ($\dot{\theta}_i(t)$) as

$$\dot{\theta}_i(t) = f_i + r_i(t) . \tag{2.16}$$

The phases of the neighboring nodes are received and compared to the local phase ($\theta_i(t)$), averaged and filtered to finally obtain a regulated waveform at the output of the VFO. The control signal can be written as

$$r_i(t) = h_i(t) * \sum_{k=1}^{N} a_{ik} (\theta_k(t - t_{ik}) - \theta_i(t)) . \tag{2.17}$$

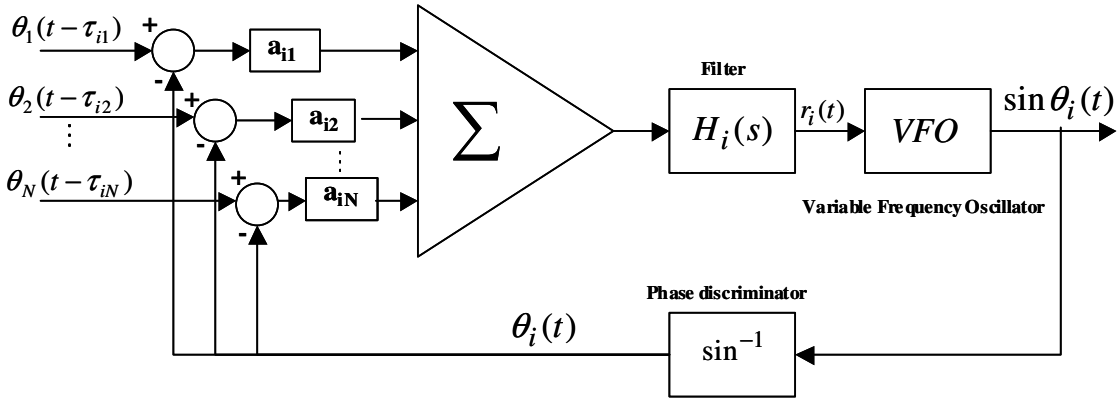

Figure 2.4 Multiple-input PLL in every node of the system in [24]

Where * is the convolution operation. Substituting (2.17) in (2.16) assuming $\sum_{k=1}^{N} a_{ik} = 1$, and taking the Laplace transform of the resulting expression yields

$$s\Theta_i(s) - \theta_i(0) = \frac{f_i}{s} + H_i(s)\sum_{k=1}^{N} a_{ik}\Theta_k(s)e^{-s\tau_{ik}} - H_i(s)\Theta_i(s). \tag{2.18}$$

Taking $V_i(s) = \frac{f_i}{s} + \theta_i(0)$, and reordering (2.18) yields

$$\Theta_i(s) = \frac{H_i(s)}{s + H_i(s)}\sum_{k=1}^{N} a_{ik}\Theta_k(s)e^{-s\tau_{ik}} + \frac{1}{s + H_i(s)}V_i(s). \qquad \forall i \in [1,\ldots,N]. \tag{2.19}$$

The set of *N* equations in (2.19) can be written in matrix form if the following matrices are defined: $\mathbf{B}(s)_{NxN}$ with ik^{th} element given by, $b_{ik}(s) = \frac{H_i(s)}{s + H_i(s)}a_{ik}e^{-s\tau_{ik}}$,

$\mathbf{C}(s)_{NxN}$ is a diagonal matrix with $c_{ii}(s)=\dfrac{1}{s+H_i(s)}$, $\mathbf{\Theta}(s)_{Nx1}$ with $\Theta_{i1}(s)=\Theta_i(s)$, and $\mathbf{V}(s)_{Nx1}$ with $V_{i1}(s)=V_i(s)$. Then

$$\mathbf{\Theta}(s)=\mathbf{B}(s)\mathbf{\Theta}(s)+\mathbf{C}(s)\mathbf{V}(s) \;\Rightarrow\; \mathbf{\Theta}(s)=[\mathbf{I}-\mathbf{B}(s)]^{-1}\mathbf{C}(s)\mathbf{V}(s). \tag{2.20}$$

The matrix $[\mathbf{I}-\mathbf{B}(s)]^{-1}\mathbf{C}(s)$ is the transfer function of the system. Taking $\sum_{k=1}^{N} a_{ik}=1$, and $H_i(0)=\lambda_i>0$ the authors in [24] found that the condition for this system to achieve stability is that the network must be connected and that

$$\left|\frac{H_i(s)}{s+H_i(s)}\right|<1,\ \forall i\in[1,\ldots,N]. \tag{2.21}$$

The steady-state frequency at the output of every oscillator will stabilize to a value that is proportional to the average of all the individual free-running frequencies.

One important parameter for the stability of the approach is the latency of the communication links. Link latency affects the validity of the timing information exchanged among the oscillators and ways to estimate and compensate for it are required in case it is not negligible.

As an example of the previous approach, assume a fully connected network of *N* oscillators with a filter $H_i(s)=K_i$ (i.e., a proportional controller). In that case $\mathbf{B}(s)_{NxN}$ and $\mathbf{C}(s)_{NxN}$ are given by

The i^{th} row of $\mathbf{B}(s)_{NxN}$ is

$$\left\{\frac{K_i}{s+K_i}a_{i1}e^{-s\tau_{i1}},\frac{K_i}{s+K_i}a_{i2}e^{-s\tau_{i2}},\ldots,\frac{K_i}{s+K_i}a_{ii-1}e^{-s\tau_{ii-1}},0,\frac{K_i}{s+K_i}a_{ii+1}e^{-s\tau_{ii+1}},\ldots,\frac{K_i}{s+K_i}a_{iN}e^{-s\tau_{iN}}\right\}$$

, and

$$\mathbf{C}(s)=diag\left\{\frac{1}{s+K_1},\frac{1}{s+K_2},\ldots,\frac{1}{s+K_N}\right\}.$$

Assume that $a_{ik}=1/(N-1)$, $\forall i,k\in[1,\ldots,N]$ for $i\neq k$, and $a_{ii}=0$, such that $\sum_{k=1}^{N}a_{ik}=1$.

The stability of this system depends on the existence of the inverse of $[\mathbf{I}-\mathbf{B}(s)]$ as can be observed from the transfer function $[\mathbf{I}-\mathbf{B}(s)]^{-1}\mathbf{C}(s)$. Note that the $[\mathbf{I}-\mathbf{B}(s)]$ matrix is diagonally dominant. In a diagonally dominant matrix the sum of the magnitudes of the off-diagonal elements of any row is smaller than the magnitude of the same-row element located in the main diagonal of the matrix. That is, if $\mathbf{C}_{NxN}$ is a diagonally dominant matrix then $|c_{ii}|>\sum_{i\neq k}|c_{ik}|,\forall i\in[1,\ldots,N]$. A diagonally dominant matrix is non-singular (see [24] for this proof), therefore we conclude that $[\mathbf{I}-\mathbf{B}(s)]^{-1}\mathbf{C}(s)$ exists and the system is stable in the sense that a bounded input will produce a bounded output. Note that the condition (2.21) is necessary to ensure $[\mathbf{I}-\mathbf{B}(s)]$ is a diagonally dominant matrix. Furthermore, a simple proportional controller can always satisfy the condition in (2.21). A proportional controller, however, is not the most ideal solution since the system will constantly need to receive timing information at its inputs in order to keep the phases of the different oscillators aligned. This is due to the skew factor explained in Section 2.1.1.

A possible solution is to incorporate a mechanism that estimates the skew difference among the oscillators in the network if the continuous transmission of a timing signal is to be avoided or minimized.

Note that it is not always possible to guarantee the connectivity of the network at all times in a wireless network. However, it is assumed throughout this work that network connectivity is present at the time the nodes exchange their timing information. Stability does not necessarily imply convergence. However, convergence is shown to be achievable in [24], and in our particular approach in Chapter 3. In the particular example above, convergence is guaranteed as long as the free-running frequencies of the oscillators (the inputs to the system) are all equal.

Our work differs from [24] in that there is no direct physical control of the clocks or oscillators in every node, which translates into an implementation advantage since there is no need to have access to the physical (PHY) layer of the system. CSMNS is a discrete NTS algorithm that utilizes the clock-sampling technique (Section 1.2.1) to exchange timing information in combination with a mutual network synchronization approach. Furthermore, we introduce simple mechanisms that can be used to enhance the synchronization performance in a wireless Ad Hoc network by taking advantage of the flat hierarchical architecture of a mutual NTS approach.

The previous mutual synchronization ideas can be used as part of an NTS approach in which the nodes of the network synchronize to a number of fixed reference nodes. Figure 2.5 shows this approach, which we call multi-reference broadcast synchronization. Every reference node (nodes *a*-*d*) sends its timing (phase) information, which is used by every node in the network to synchronize. Each reference node will mutually synchronize to the

other reference nodes. The whole network is covered by the combined broadcast of all the reference nodes. Each reference node could be fixed or mobile and connected to an "unlimited" source of energy (e.g., an automobile or airplane).

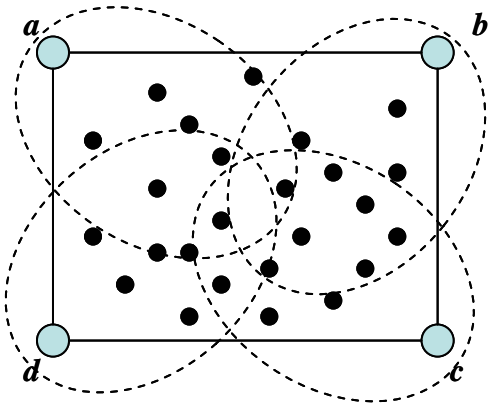

Figure 2.5 A mutual NTS with fixed broadcasting nodes

Additionally, the reference nodes can be connected through reliable communication links with one another. A network formed in this way is a clustered network. The advantage over a purely master-slave approach is that of resilience to node and link failures. In case of a link failure, a given reference node can always extract the timing information from the other remaining reference nodes automatically and without performing any specific procedures since there is no hierarchical structure. In case of a node failure due to, for instance, an attack or a malfunction, the rest of the reference nodes can detect this condition and simply increase their power to compensate for the signal lost. Note that, from the non-reference nodes point of view, the synchronization is a master-slave approach. Furthermore, the reference nodes can be used for information multicasting or broadcasting in addition to the synchronization purpose. CSMNS differs

from the latter approach in that the same nodes that are part of the network are responsible for their own synchronization. However, an approach like the one in Figure 2.5 can be a useful alternative in some scenarios.

A mutual network synchronization approach is proposed for a wireless Ad Hoc network of automobiles in [25]. However, no study is made of the performance of this approach in a multi-hop network, and the timing information is exchanged using very short pulses that can occupy a large bandwidth. Additionally, this implies the use of special circuitry and tight restrictions on the Rx-Tx and Tx-Rx turn-around times of the radios if a half-duplex scheme is used, otherwise, full-duplex radios are needed to simultaneously listen and send the continuous train of pulses.

Hu and Servetto [20] proposed a scheme for sensor networks in which a centrally located sensor initializes the NTS approach by sending pulses that are relayed by neighboring nodes in a form of hierarchical flooding. This approach inherits the PHY layer requirements from the earlier work in [24] and [25] through the use of a continuous train of pulses. Moreover, it is a hierarchical approach since it depends more on the central sensor than on any other sensor of the network. In particular, it is unclear how to select and discover the centrally located sensor, or how to replace it in case it fails or its battery power is drained beyond a point where it can no longer transmits pulses. In most future Wireless Sensor Networks the deployment of the sensors will not be engineered, therefore the relative location of the sensors will be random. Consequently, those approaches that do not rely on specific nodes located in strategic places will be preferred. Additionally, methods that do not rely on the continuous transmission of information will have the advantage of being more energy conscious.

The Reference Broadcast Synchronization (RBS) scheme is proposed with multi-hop support in [21]. RBS achieves the synchronization of multi-hop neighborhoods through the exchange of messages with an intermediate node (i.e., a node in between broadcasting neighborhoods). The purpose is to obtain a logical ordering of events rather than achieving real synchronization of the clocks in a multi-hop scenario. In RBS, the network is divided into broadcasting neighborhoods in which a single "reference node" sends packets that are used to achieve local synchronization (i.e., the reference node's neighborhood). One of the fundamental ideas introduced by the creators of RBS is that of receiver-based synchronization. A receiver-based synchronization approach exploits the fact that packets sent by the reference node (or master clock) arrive at approximately the same time in every node of its neighborhood. After the packet is sent, the receiving nodes exchange the times at which everyone received it in order to achieve synchronization. In particular, assume there are N nodes in the neighborhood served by a reference node. If we use similar notation to the one in Figure 2.2, the i^{th} node receives the reference packet (not necessarily carrying a timestamp) at $T_{C_r}^i$ according to its local clock. Each node transmits its T_{C_r} to the other nodes. Therefore, every node has information of its offset with respect to the rest of the nodes. A possible way to compute a corrected value of time could be given by

$$T_i(n) = \max_j \{T_{C_r}^j\} \quad j \neq i, \text{ and } \forall i, j \in [1,...,N] \tag{2.22}$$

Note that averaging the differences between the set of T_{C_r} times could result in clocks moving backward, which is not a desirable property when the synchronization is used to

order events in a distributed system. The important fact, according to the authors, is that the time-uncertainties on the server side, explained at the beginning of this section in the context of client-server NTS approaches, are eliminated. Specifically, the exact time of the timestamp transmission on the server is unimportant as long as all the receiving nodes receive the packet at the same time. However, certain problems remain in RBS.

Firstly, RBS is a hierarchical approach since it depends on a reference node for the successful operation of the approach. A hierarchical approach might be suitable for networks in which the nodes barely move relative to one another. However, it is not the most suitable approach for highly dynamic wireless Ad Hoc networks with mobility. An important limitation is the overhead created by the need of the receiving nodes to exchange their τ_{C_r} times. In a neighborhood of *N* nodes this overhead can be in the order of $O(N^2)$. Additionally, it is not clear how to synchronize the reference node to the rest of the network if needed.

The fundamental problem attacked by RBS through a receiving-based synchronization method can be approached in a different way. Our approach is to exploit the fact that the timing information in a mutually synchronized network is not derived from a central or reference clock. It is unimportant which node sent the timestamp in a mutual NTS approach since the timing information is extracted from *all* the clocks in the network. Assuming all the nodes contend to send their timestamps in a contention time-window, it is obvious that in a hierarchical approach the most important node (e.g., the reference node) will not know exactly when its timestamp will get transmitted during this contention window. This is the major source of uncertainty attacked by RBS. However, in a hierarchically flat approach, what is important is when the node *plans* to transmit its

timestamp rather than when *it is* transmitted. If a node is unsuccessful in its attempt to send its timestamp at the planned time (because the medium was busy), it can simply refrain from doing so and let other nodes transmit, in this way the timestamp transmission uncertainty can be considerably reduced or eliminated. Note that the latter procedure is used by the TSF in the IEEE 802.11 standard. However, the TSF does not use a mutual synchronization approach, therefore the latter procedure does more harm than good as will be explained in more detail in the next section. Other numerous NTS approaches have been proposed for Wireless Ad Hoc and sensor networks, see for instance [26]-[36].

2.3 The IEEE 802.11 Timing Synchronization Function

The TSF of the IEEE 802.11 standard is a network synchronization algorithm that utilizes the clock-sampling method to exchange timing information. The main goal of the TSF is to synchronize the time shown by the physical clocks in the network, referred to as timers in the standard, in order to support the power management operation in all PHY layer versions, and the channel hopping procedure in the frequency hopping spread spectrum PHY layer. The TSF utilizes many of the ideas presented in [22]. In particular, it tries to fulfill the same principle of clock correctness introduced in [22] through the use of physical clocks and timestamp messages.

In this section we present a description and analysis of the TSF when the network is configured to operate in an Independent Basic Service Set (IBSS) mode, commonly known as “Ad Hoc” mode. The Ad Hoc mode of IEEE 802.11 defines a wireless local area network (LAN) of *stations* (nodes) that do not share a central point of management

or control. An IBSS is an infrastructureless wireless LAN in which every node can communicate with one another in a single-hop manner through the wireless medium. The IEEE standard also defines an infrastructure mode in which there is a central entity called the access point that is responsible, among other things, of the forwarding of data packets from one node to another. If a node wants to send a message to another node, it has to do so through the access point. We focus our attention exclusively on the NTS method used in the Ad Hoc mode, and call it simply the TSF.

The TSF method is described as follows,

1. Each node sends a beacon periodically at a Target Beacon Transmission Time (*TBTT*) with period *aBeaconPeriod* (e.g., 0.1 sec [2]). At each *TBTT* each node shall:
2. Suspend the back-off timer of any pending non-beacon transmission.
3. Calculate a random delay uniformly distributed in the range between zero and $2 \cdot aCWmin \cdot aSlotTime$. Table 2.1 shows the values of *aCWmin* and *aSlotTime* for the IEEE 802.11 standard with different PHY layer versions (i.e., Frequency Hopping, Direct Sequence Spread Spectrum, and Orthogonal Frequency Division Multiplexing).
4. Wait for the period of random delay before transmitting the beacon.
5. Cancel the remaining random delay and the pending beacon transmission, if a beacon arrives before the random delay timer has expired.
6. Send a beacon if the random delay has expired and no beacon arrived during the delay period. A beacon will be transmitted only if the medium is sensed to be idle.

Table 2.1 Beacon contention window parameters used in IEEE 802.11

Parameter	FHSS	DSSS	OFDM
aCWmin	15	31	15
aSlotTime	50	20	20

Upon reception of a beacon a node will adjust the received timestamp to take into account its physical layer delay. The receiving node will set its clock to the value of the adjusted timestamp if it is later than the local timestamp. The TSF clock is a 64 bit counter with 1 μsec resolution. Figure 2.6 depicts a time-diagram of the TSF. Every station (node) contends to transmit its timestamp beacon within a contention window that starts at a *TBTT*. If the medium is busy a node defers its beacon transmission for the next *TBTT,* and it transmits its beacon if the medium is idle and it has not received a beacon from another node.

An approximate analysis of TSF in the single-hop case was first attempted in [37]. The probability of sending one beacon successfully regardless of the node that sent it (P_{any}), and the probability of sending a beacon successfully by a given node (P_{given}) were found under the assumption that perfect synchronization has been achieved. That is, the beacon contention window of every node starts at the same time for all the nodes in the network. However, the analysis proves the inefficiency of TSF to scale even to moderate number of nodes.

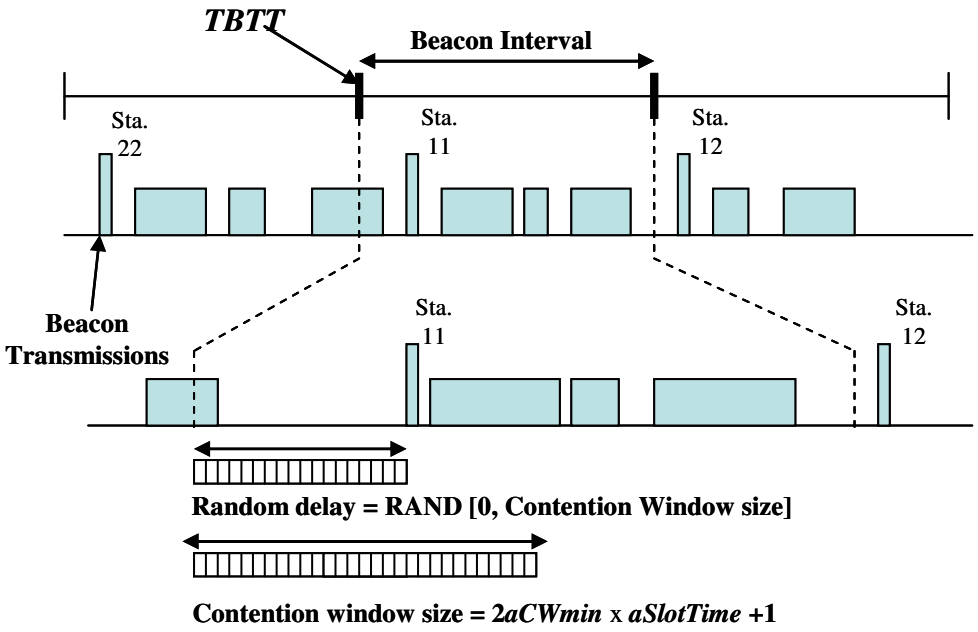

Figure 2.6 Time-diagram of the TSF

The reason for the lack of scalability of the TSF is blamed on the beacon collisions, which make P_{given} small as the number of nodes in the network increases. Figure 2.7 shows P_{given} versus the number of nodes in a network. The parameters used in Figure 2.7 correspond to those used in an IEEE 802.11 FHSS network (see Table 2.1). The beacon transmission takes 11 slots (i.e., 550 $\mu secs$). As seen in Figure 2.7, for a network of 20 nodes the probability of a given node to transmit its beacon is approximately 0.05. This is equivalent to stating that the probability of receiving a beacon from the node with the fastest clock is 0.05 when the number of nodes in the network is equal to 20. The low probability of sending a beacon successfully by the node with the fastest clock translates into severe a-synchronism when the clocks of the network drift with respect to one another. This affects power management, and the channel hoping procedure in the IEEE

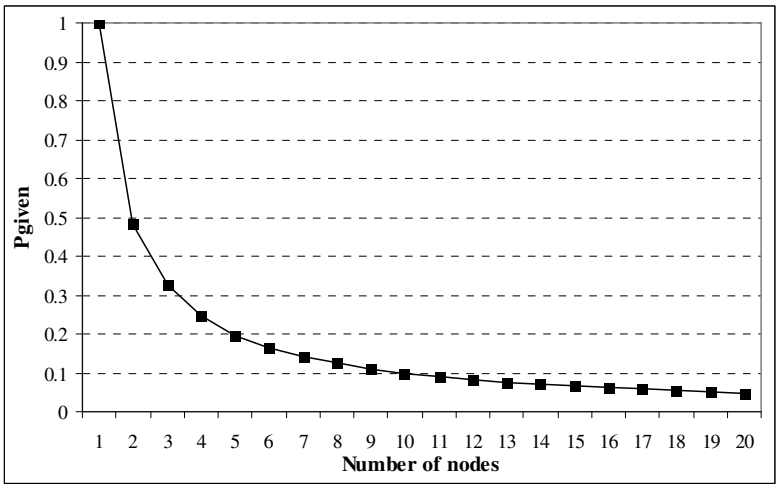

Figure 2.7 P_{given} **for the IEEE 802.11 TSF**

802.11 standard, furthermore, it proves the lack of scalability of the TSF algorithm. At the core of this problem is the fact that the TSF depends, more than anything else, on the successful transmission of beacons by the node with the fastest clock in the network. However, the TSF does not make any attempts to ensure that this happens.

The authors in [37] realized the latter and proposed to simply give more priority to the node with the fastest clock. However, finding the fastest clock is not a trivial task, particularly in highly dynamic networks where nodes continuously leave and join the network.

A simple way to try to improve the TSF is to allow beacon transmissions even after successfully receiving a beacon (hereafter called secondary beacon transmissions). That is, modify steps 5 and 6 above in the description of TSF in order to allow a node to transmit its beacon even after successfully receiving one. This approach is not the most ideal however, since we are likely improving the TSF at the expense of increasing overall network energy consumption and overhead. Energy consumption will be increased since

more beacons will be transmitted on average, and overhead increases since it is more likely that the contention window will be extended further by the secondary beacon transmissions. A larger beacon contention window implies smaller bandwidth for actual data transmission. However, this method might prove to be useful in cases in which even a little improvement in synchronization would be worth the cost.

We call the previous procedure the modified TSF and it is analyzed next extending the analysis in [37]. It is assumed that collision of beacons is the only cause for errors in their reception, and that a collision, even if it is partial, causes the destruction of the beacon. The probability of a given station to transmit its beacon in the modified TSF ($\hat{P}_{given}$) is given by

$$\hat{P}_{given}(n,W) = \frac{1}{W+1}\sum_{k=0}^{W} \hat{P}^{k}_{given}(n,W,k)\,. \tag{2.23}$$

Where $\hat{P}^{k}_{given}(n,W,k)$ is the conditional probability that the given node successfully transmits a beacon given that it is scheduled to transmit in slot k; $W+1$ is the contention window size (there are $W+1$ slots labeled 0 through W), and n is the number of nodes in the network. $\hat{P}^{k}_{given}(n,W,k)$ can be computed based on the same events outlined in [37] plus an additional one allowing one node to transmit even after a successful beacon reception. $\hat{P}^{k}_{given}(n,W,k)$ is given by

$$\hat{P}^{k}_{given}(n,W,k) =$$

$$\begin{cases} \left(\frac{W-k}{W+1}\right)^{n-1} & \text{for } k<b, \forall n \text{ or } k \geq b, n=1 \\ \left(\frac{W-k}{W+1}\right)^{n-1} + \sum_{i=0}^{k-b}\left\{\left(\frac{1}{W+1}\right)\cdot \hat{P}^{k}_{given}(n-1,W-i-b,k-i-b)\right\} & \text{for } k \geq b, n=2 \\ \left(\frac{W-k}{W+1}\right)^{n-1} + \sum_{i=0}^{k-b}\sum_{x=1}^{n-1}\sum_{y=0}^{n-1-x}\left\{C_x^{n-1}C_y^{n-1-x}\left(\frac{1}{W+1}\right)^{x}\left(\frac{b-1}{W+1}\right)^{y}\right. & \\ \qquad \left.\cdot\left(\frac{W-i-b+1}{W+1}\right)^{n-1-x-y}\hat{P}^{k}_{given}(n-x-y,W-i-b,k-i-b)\right\} & \text{for } k \geq b, n \geq 3 \end{cases} .$$

(2.24)

The boundary conditions of (2.24) are $\hat{P}^{k}_{given}(0,W,k) = \hat{P}^{k}_{given}(n,0,k) = (\hat{P}^{k}_{given}(n,W,k)$ $\forall\, W<k) = 0$, which express the fact that $\hat{P}^{k}_{given}(n,W,k)$ is equal to zero in the extreme cases in which the number of nodes is zero, the contention window size is zero and the contention window size is smaller than the slot chosen for transmission respectively. $\binom{n}{x} = \frac{n!}{x!\cdot(n-x)!}$ is the binomial coefficient. A beacon is assumed to take b slots to transmit. Additionally, $W>b$ is implicitly assumed, otherwise there will not be enough time to transmit even a single beacon. Equation (2) is the union of three disjoint events characterized by the relative position of the k^{th} slot and the number of nodes in the network. Three different scenarios cover all possibilities. The first expression in (2) considers the case when $k<b, \forall n \text{ or } k \geq b, n=1$, the second expression when $k \geq b, n=2$, and the third expression when $k \geq b, n \geq 3$.

In the first expression $\hat{P}^{k}_{given}(n,W,k) = \left(\frac{W-k}{W+1}\right)^{n-1}$ because the successful transmission of the given node (starting at the k^{th} slot) for $k<b, \forall n$ depends on the probability that all

remaining $n-1$ nodes will transmit after the k^{th} slot, otherwise the transmission of the other node(s) will cover the k^{th} slot due to the fact that a beacon transmission takes b slots. If $k \geq b$ and $n=1$, then $\hat{P}^{k}_{given}(n,W,k)=1$, which is also characterized in the same expression.

If $k \geq b$ and $n=2$, then it is required that a single successful transmission, or no transmissions occurred before the k^{th} slot for the transmission of the given node to be successful. Note that the former event is only valid in the extended TSF since normally a node will defer its beacon transmission for the next *TBTT* if a beacon is received before. The second expression in (2) considers these two events. $\left(\frac{W-k}{W+1}\right)^{n-1}$ is the probability that the rest of the nodes will transmit after the k^{th} slot, or in other words, that the given node is successful because no beacon transmission occurred before. $\sum_{i=0}^{k-b}\left\{\left(\frac{1}{W+1}\right)\cdot \hat{P}^{k}_{given}(n-1,W-i-b,k-i-b)\right\}$ is the probability that the given node is successful transmitting after a single successful transmission that occurred at slot $i<k$. Slot i must be in the range between zero and $k-b$, otherwise it will cover the k^{th} slot, therefore $i \leq k-b$. The latter explains the limits of the summation in the second expression. The expression inside the summation is comprised of two terms, the probability of picking slot i (i.e., $\left(\frac{1}{W+1}\right)$) and the probability of successful transmission of the given node expressed in a recursive way. That is, after subtracting the previous successful transmission at slot i and taking into account the time it takes to transmit a beacon (i.e., $\hat{P}^{k}_{given}(n-1,W-i-b,k-i-b)$).

The third term is for $k \geq b$ and $n \geq 3$. In this case the given node is successful if there are no transmissions before the k^{th} slot (i.e., $\left(\frac{W-k}{W+1}\right)^{n-1}$), or if exactly x beacon transmissions occurred in slot $i \leq k-b$, where $(1 \leq x \leq n-1)$, and that exactly y nodes $(0 \leq y \leq n-1-x)$ are scheduled to transmit in slots $i+1$ through $i+b-1$. The nodes scheduled to transmit during the latter interval will defer their transmissions for the next *TBTT* due to the beacon transmission that started at slot *i* (errorless carrier sensing is assumed). Therefore before the k^{th} slot, a total of $x+y$ nodes are scheduled to transmit, but none of the actual transmissions cover the k^{th} slot. It is clear that collisions will occur if $x>1$, but the given node will transmit successfully as long as these transmissions do not cover the k^{th} slot. The second term of the third expression has an analogous explanation to the second term of the second expression. Note that the second expression is a special case of the third when $x=1$, $y=0$, and $n=2$.

Equation (2.23) is plotted in Figure 2.8 along with P_{given} of the TSF and simulation points of the procedure performed by the modified TSF. The simulation points were obtained after 30 *minutes* of real-time simulation using Matlab. As can be seen, the difference between Equation (2.23) and the simulated results is negligible. The first thing to notice about the modified TSF is that, as expected, it achieves a better probability of beacon transmission than TSF, however, it still suffers from severe degradation when the number of nodes in the network increases.

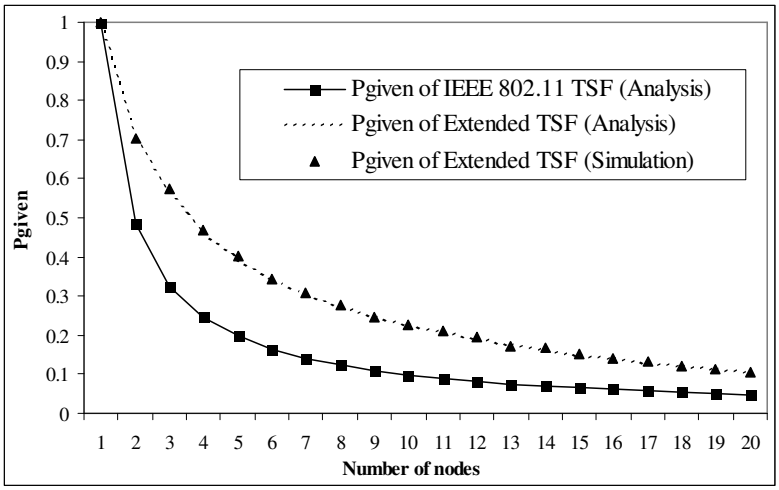

Figure 2.8 P_{given} **of the TSF and the modified TSF**

Therefore, although to a lesser degree, the modified TSF suffers from the same scalability problems of the original TSF. One could try to further improve the modified TSF and allow secondary beacon transmissions only from those nodes that had a larger local timestamp than the timestamp received. We performed simulations of this approach in a network of 10 and 20 nodes in which the fastest clock drift at +25ppm (i.e., gains 2.5 μ*secs* with respect to real time every *aBeaconPeriod* = 0.1 *secs*), and the rest of the clocks drift at –25ppm, with the FHSS parameters.

Figure 2.9 shows the cumulative distribution function (c.d.f) of the maximum time difference among the nodes using this *new* modified TSF. The new modified TSF achieves better performance since the maximum time difference is smaller; however, it is still unsatisfactory because the small accuracy is gained at the expense of more beacon transmissions.

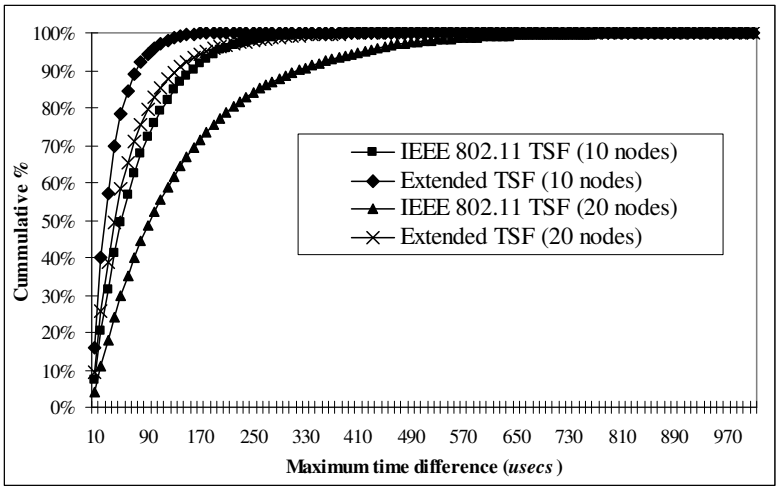

Figure 2.9 c.d.f of the accuracy in the TSF and new modified TSF with 10 and 20 nodes

Note that the accuracy of the modified TSF with 20 nodes is roughly the same as the original TSF with 10 nodes (Figure 2.9). This is approximately what we observe in Figure 2.8 if a horizontal line is drawn from the point of 10 nodes in the TSF curve, to the intersecting point with the curve of the modified TSF.

The probability P_{any}, defined earlier, can give an indication of the potential improvement of a mutual NTS algorithm over a more hierarchical approach such as the TSF. Figure 2.10 shows an analytical result of P_{any} obtained in [37] over our own simulation result for two different PHY layers (i.e., DSSS and FHSS). Also shown is P_{given}. Figure 2.10 suggests that a mutual network synchronization algorithm based on all the beacons transmitted in the IEEE 802.11 standard can greatly improve over the TSF. That is, if the timing information embedded in *every* beacon transmitted is used, then the performance improvement would be substantial since the probability of any beacon to be

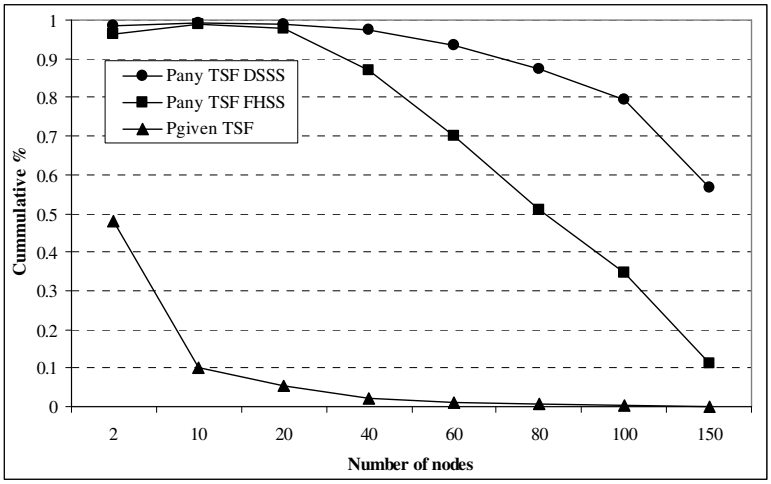

Figure 2.10 P_{any} **and** P_{given} **for the TSF**

successfully received is much larger than the one for a specific beacon (e.g., coming from the fastest clock). For instance, with 100 nodes and DSSS PHY layer parameters, the probability of sending a beacon by any node is approximately 80%, while P_{given} is almost zero. This points out that our efforts should be focused on non-hierarchical NTS approaches that exploit the information carried by every beacon transmitted.

Another important drawback of the TSF is that it is marginally convergent. It requires that the nodes transmit their beacons regularly as long the network exists. The TSF does not incorporate any learning mechanism that can be used to adaptively enlarge the time difference between two *TBTT*s, even if the nodes are static and the wireless links have good reliability. To see this, we have performed simulations of the TSF in a multi-hop network with a regular-structure topology. The topology of the nodes is shown in Figure 2.11. It is a square-grid in which the number of nodes in the network is $N = n^2$. The lines

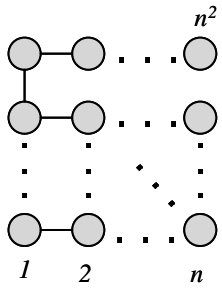

Figure 2.11 Topology used for the multi-hop simulation

joining the nodes represent the transmission ranges of the nodes. The transmission range is taken to be half the detection range. The latter means that a receiving node is only able to decode a beacon properly if it is received within the *transmission* range of the transmitting node (not counting wireless medium impairments and collisions). However, a collision can happen if a third node transmits within the *detection* range of the receiving node.

The simulations were performed assuming no capture and clock models given by (2.10). The no-capture assumption implies that a receiver is not able to properly discern between two overlapping beacons in time, both beacons are considered lost. A constant 1% beacon error rate is assumed to model wireless medium impairments. That is, a beacon might not be counted as properly received with 1% probability, even if it did not collide with other beacon(s). Mobility of the nodes and more realistic wireless medium impairments (i.e., small-scale and large scale fading) are not considered in the simulations since it is not our main goal to obtain absolute measures of performance, but rather insights into fundamental limitations and advantages. More detrimental conditions in the wireless medium can only make the performance of any NTS approach worse. The

time it takes to transmit a beacon was taken to be 11 slots, and *aBeaconPeriod* = 0.1 *secs*. The clocks are modeled using (2.10) with skews that take either a value of -25ppm, or +25ppm. The hidden and exposed node problems are taken into account in the simulation. An exposed node will defer its beacon transmission if another node transmits within its detection range; a hidden node will cause a collision, if it is within transmission range of a receiving node, but out of the detection range of the corresponding transmitting node. The total simulated time is 30 *minutes*.

Figure 2.12 shows the c.d.f of the maximum time difference ($T_{\max}$) in an $n = 5$, or 5x5 network, where $T_{\max} = \max_{\forall i \neq j}\{|T(i) - T(j)|\}$. Node 1 has the fastest clock with a skew equal to +25ppm, and the rest of the nodes skew at -25ppm. The purpose of the latter is to observe the performance of the TSF when there is a single faster clock and the remaining clocks are all slower, this is a worst case scenario for the TSF. The maximum value of $T_{\max}$ is approximately 900 $\mu secs$, with a sample average of 360 $\mu secs$.

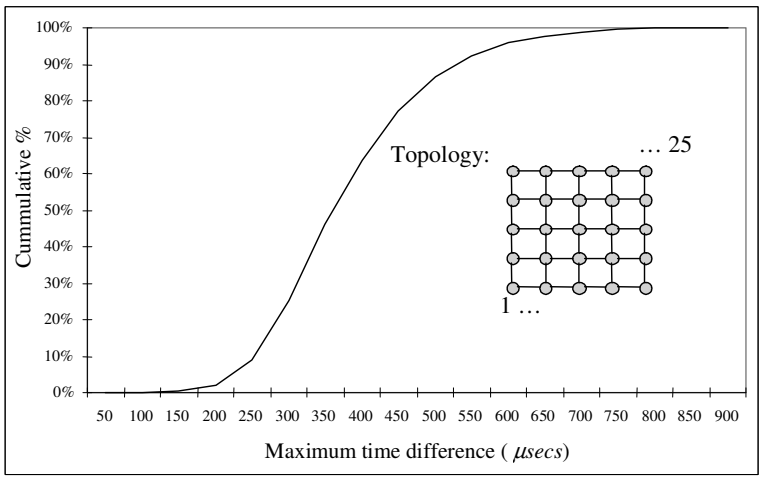

Figure 2.12 $T_{\max}$ c.d.f for the TSF in a 5x5 network topology

A snapshot of the time difference between nodes 1 and 7, and nodes 1 and 25 is shown in Figure 2.13 for the first 100 *secs* of simulation. Note that even under no mobility the TSF requires the continuous exchange of beacons in order to keep the nodes synchronized.

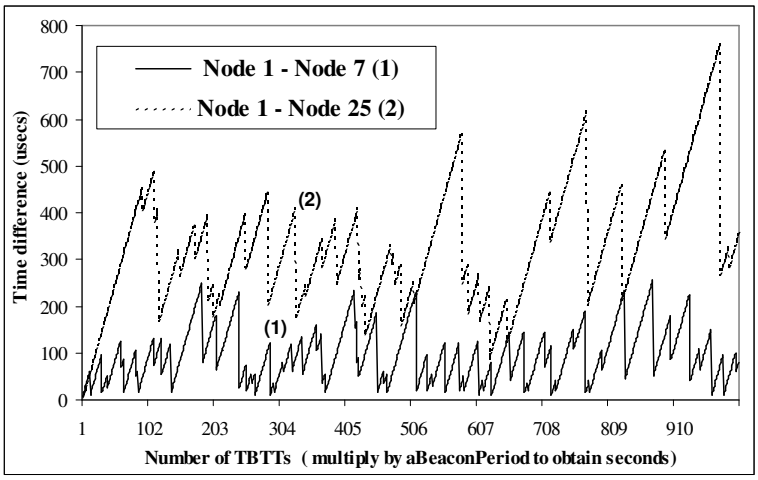

Figure 2.13 Time difference between nodes 1-7 and nodes 1-25 during 30 *minutes* of real-time (TSF)

When a node receives a timestamp with a later time than its own local time the time difference with respect to the fastest clock decreases, this is observed when the curve drops in Figure 2.13. However, the difference will start to increase again as long as the given node does not receive a later timestamp. The 1-7 time difference is smaller due to the closer proximity of node 7 to node 1. Node 1 is the node with the fastest clock, and therefore the most important node from the TSF point of view. Two important conclusions can be drawn from these results. First, the TSF is marginally convergent, and second it does have the potential to synchronize multi-hop networks albeit with an average accuracy in the order of hundreds of micro-seconds at best. To the best of our

knowledge, this is the first published result of the TSF performance in a multi-hop scenario [38].

2.4 Summary

In this chapter we have presented concepts on network time synchronization that are needed to motivate and understand our proposed network synchronization algorithm. We have focused on the mutual network synchronization concept and the IEEE 802.11 TSF. The former is the fundamental idea behind our proposed algorithm, and the latter is a standardized network synchronization algorithm that will be use in the next chapter as benchmark for comparison.

Based on the previous analysis, we conclude that a synchronization algorithm that truly improves over the TSF should:

1. Abandon the idea of giving the responsibility of network synchronization to a node with particular characteristics (e.g., fastest clock, node with more degree in the network, centrally located etc). This will improve the chances of spreading the timing information, increase the robustness of the algorithm to network dynamics, and increase the speed of convergence of the algorithm.
2. Take full advantage of the exchanged timestamps in order to have a larger probability of spreading the timing information over the network.
3. A NTS approach should be convergent, in the sense that the maximum time error between any pair of clocks in the network should try to approach zero as much as the

resolution of the clocks and the dynamics of the network allow. A convergent NTS approach will help in the implementation of a mechanism that adapts the amount of overhead needed to achieve synchronization in an opportunistic manner. For instance, the period necessary to transmit a beacon can be made inversely proportional to the time error between the clocks.

The latter goals should be achieved with small increase in energy expenditure and overhead over the already standardized algorithm. We show that the algorithm presented in the next chapter achieves all these goals plus the ones presented in Section 1.4.

Chapter 3

Clock Sampling Mutual Network Synchronization

This chapter presents the proposed NTS method referred to as Clock Sampling Mutual Network Synchronization (CSMNS). To the best of our knowledge, CSMNS is the only NTS approach proposed for wireless Ad Hoc networks that leverages on the concept of mutual network synchronization. First, the basic model of CSMNS is presented and the equations are numerically solved for a single-hop scenario in a fully connected network. Second, a generalization of CSMNS is presented, with the purpose of reducing overhead, referred to as CSMNS Rotating Master Node (CSMNS-RMN). Simulation results of the CSMNS algorithm and variants are presented at the end of this chapter in multi-hop and single-hop scenarios along with a discussion of the results and the network architectures for which CSMNS is deemed to be more suitable.

3.1 The CSMNS Algorithm: Description and Analysis

The CSMNS algorithm is based on the mutual network synchronization approach presented in Chapter 2. The CSMNS algorithm can be described as follows:

1. We define a controlled clock and a real clock in each node. The real clock could be the same timer used in the TSF (64 bits @ $1\mu sec$ resolution). The controlled clock reads from the real clock and adjusts the value read by a correction factor we denote as $\boldsymbol{s}$. Synchronization information for any purposes (e.g., management, security, MAC support, space-time event relationships etc) is taken from the controlled clock. We see $\boldsymbol{s}$ as a control parameter that adjusts the speed of the controlled clock. If $\boldsymbol{s} = 1$, the controlled clock is no different from the real clock except for a negligible difference caused by the processing time of $\boldsymbol{s}$. Figure 3.1 shows the relationship between the controlled and real clocks. Without loss of generality we assume that the physical layer and processing delays are taken into account and the controlled timestamp has been adjusted accordingly if necessary. A node must scan beacons for some period of time in order to acquire synchronization before joining the network. The node listens for beacons and sets the timestamps of the controlled and real clocks to the value of the timestamp received. The value of $\boldsymbol{s}$ is set to 1 at initialization. The requirement to acquire some information at the beginning of a session when a node enters a network is common for other protocols and other types of networks, such as routing protocol information in wireless Ad Hoc networks or system information parameters in cellular radio networks. It is, for instance, a requirement in the IEEE 802.16a standard [3] to acquire coarse synchronization at initialization. We assume that initial coarse synchronization of the nodes has been achieved before the main algorithm is run.
2. All nodes contend at specific intervals of time in order to send their beacons with their controlled timestamps.

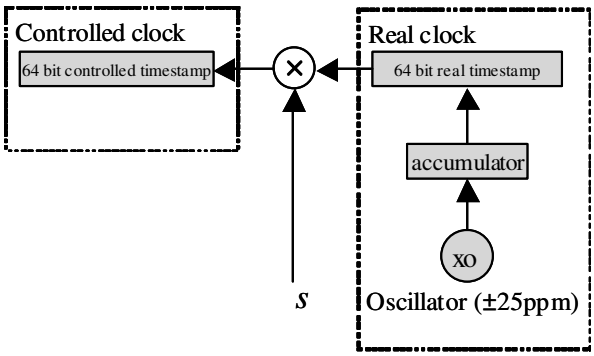

Figure 3.1 High level view of CSMNS (The numbers shown are from IEEE 802.11)

3. If the i^{th} node successfully receives a beacon, it will adjust $\boldsymbol{s}$ based on the error e_i computed as the difference between the received timestamp and the i^{th} node controlled timestamp. The value of $\boldsymbol{s}$ for the i^{th} node at the n^{th} TBTT is then computed. A more detailed description follows.

Assume a network of *N* nodes, each with a clock that, in general, has a different skew and initial time. This will result in a set of *N* equations of the following form

$$T_i(t) = \beta_i t + T_i(0),\ \ i = \{1, 2, ..., N\}. \tag{3.1}$$

The goal is to synchronize all the clocks in such a way that after some time t_c, $\left|T_i(t > t_c) - T_j(t > t_c)\right| \le \Delta T, \forall\ i \ne j.$ where t_c is the convergence time of the NTS algorithm and ΔT is the tolerable time-error. The different β_i in (3.1) make all the time processes $T_i(t)$ diverge as $t \to \infty$. The main goal of CSMNS is to minimize the relative

drifts of the time processes in (3.1). This is achieved by multiplying every time process by a correction factor $s_i(t)$; transforming (3.1) into

$$\overline{T}_i(t) = s_i(t)\beta_i t + s_i(t)T_i(0),\ \ i = \{1,2,...,N\}. \tag{3.2}$$

The correction factor can be computed in every node based on the difference between the timestamp of a received beacon and the timestamp of the local node.

Node i contends to send its time process $\overline{T}_i(t)$ ($\forall i \in [1,2,...,N]$) in periodic beacon transmissions in the same way as the IEEE 802.11 TSF. $s_i(t)$ can be written in the following discrete form

$$s_i(nT) = s_i(nT-1) + K_p \frac{(\overline{T}_{rx\_timestamp}(nT-1) - \overline{T}_i(nT-1))}{\overline{T}_i(nT-1)}. \tag{3.3}$$

Where T is the sampling period (e.g., equal to *aBeaconPeriod* in the TSF), K_p is the proportional gain, and $\overline{T}_{rx\_timestamp}$ is the timestamp of the node that successfully transmitted the beacon. The proportional gain K_p, if chosen appropriately, can average the time processes of all the clocks in the network to achieve synchronization. Substituting (3.2) into (3.3) and normalizing the sample time yields

$$s_i(n) = s_i(n-1) + K_p \frac{(s_j(n-1)\beta_j + s_j(n-1)T_j(0) - s_i(n-1)\beta_i - s_i(n-1)T_i(0))}{s_i(n-1)\beta_i + s_i(n-1)T_i(0)}. \qquad (3.4)$$

The *j* and *i* sub-indexes identify the nodes that transmitted and received the beacon respectively. Note that $s_j(n) = s_j(n-1)$. Equation (3.4) is a non-linear stochastic difference equation that can be solved numerically in $s_i(n)$ given the initial conditions $s_i(0) = 1$ and $T_i(0)$. Note that the control-law of CSMNS is similar to a proportional controller that tracks the different time processes in the network. Recall that a proportional controller meets one of the conditions for stability established in [24] for a mutual NTS approach. P_{any}, as defined in Section 2.3, is needed in order to determine whether any node successfully transmitted a beacon or not. That is, node *j* is randomly selected based on P_{any}. A binary random process with probability of success equal to P_{any} is first used to determine whether a successful beacon transmissions occurs, and then one of the nodes is selected randomly. Using the analytical result of P_{any} [37], we averaged one thousand solutions of (3.4), and found the estimate of the ensemble average of the maximum time difference ($T_{\max}$, see Section 2.3). Figure 3.2 shows $E\{T_{\max}\}$ in a network with IEEE 802.11 FHSS PHY layer parameters. $T_i(0)$ is uniformly distributed in the range [0,100], and the skews β_i are also uniformly distributed in the range $\pm\frac{25}{1e6}T$. Note the increase in convergence time with the increase in the number of nodes. The same initial transient behavior is observed in the simulations presented in the next section.

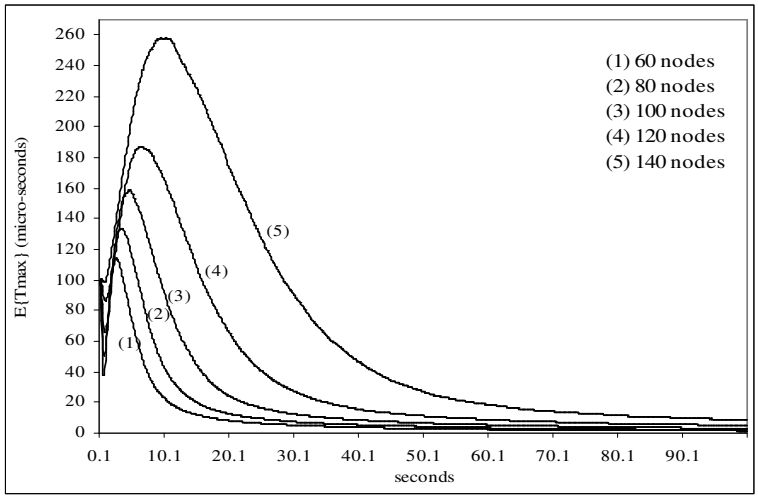

Figure 3.2 $E\{T_{\max}\}$ **of the basic-CSMNS algorithm for different network sizes**

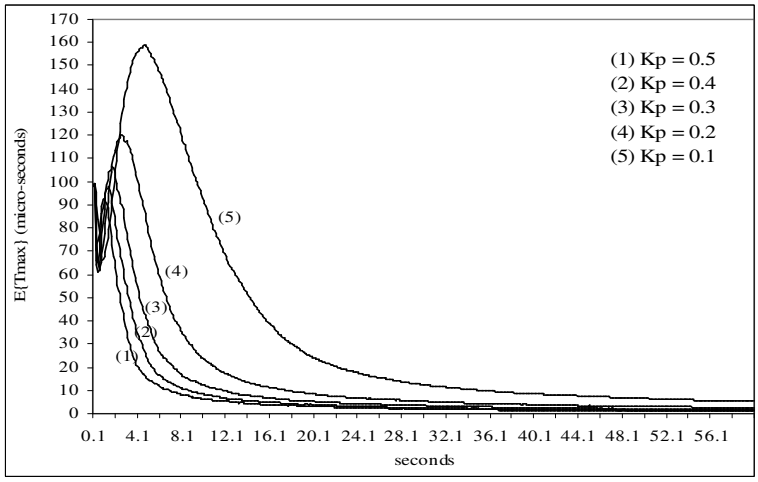

Figure 3.3 $E\{T_{\max}\}$ **of the basic-CSMNS algorithm for different** K_p

The effect of K_p on the convergence of the algorithm was also obtained. Figure 3.3 shows $E\{T_{\max}\}$ for $0.1 \le K_p \le 0.5$ in a FHSS network of 100 nodes. The algorithm presents its fastest convergence time at approximately $K_p = 0.5$. At $K_p < 0.1$ the solution seems to become unbounded. A more formal stability analysis is complicated due to the random

behavior of the system imparted by the beacon access mechanism. The values of K_p for which responses that seemed unbounded were observed depends, among other things, on the initial conditions, with a smaller initial condition contributing to a bounded response. Small initial conditions can in practice be guaranteed if any node synchronizes to the time of the network before fully participating in the network activities, by for instance, adjusting its clock based on the beacons received from neighboring nodes. An error will always be present, which explains the reason we added an initial random time-offset ($T_i(0)$) in the previous results.

Note that if an updated value of $\boldsymbol{s}$ is smaller than the previous value, the new controlled timestamp will be smaller than the previous timestamp during a short period of time. In some applications this is not desirable. A solution to this problem is to increase the real timestamp before performing the multiplication by the new smaller value of $\boldsymbol{s}$. Assume the value of $\boldsymbol{s}$ is updated from s_{old} to a new smaller value s_{new} just after receiving a new beacon. The time span that needs to elapse before the controlled timestamp shows a greater than or equal value to the previous one is given by

$$T_{\text{interval}} \geq \frac{\overline{T}_i}{s_{new}} - \frac{\overline{T}_i}{s_{old}}. \tag{3.5}$$

Where $\overline{T}_i$ is the value of the timestamp in the controlled clock when s_{new} is computed. The first ratio in (3.5) is the value that the real timestamp must have in order to obtain the same controlled timestamp after being multiplied by s_{new}, and the second ratio is the value of the real timestamp right before the change to s_{new}. Therefore, Equation (3.5) is

the time that has to elapse before the controlled timestamp reaches its original value. Using Equation (3.3) we have

$$s_{new} = s_{old} + \frac{K_p\left(\overline{T}_{rx\_timestamp} - \overline{T}_i\right)}{\overline{T}_i}, \tag{3.6}$$

Substituting (3.6) in (3.5) and after some algebra we obtain

$$T_{\text{interval}} \geq \frac{-K_p\left(\overline{T}_{rx\_timestamp} - \overline{T}_i\right)}{s_{new}s_{old}}. \tag{3.7}$$

Note that $s_{new} < s_{old} \Rightarrow \overline{T}_i > \overline{T}_{rx\_timestamp}$, which implies that T_{interval} in (3.7) is positive. Also, using $s_{new} \approx s_{old} \approx 1$, (3.7) can be simplified to

$$T_{\text{interval}} \geq -K_p\left(\overline{T}_{rx\_timestamp} - \overline{T}_i\right). \tag{3.8}$$

The time during which the controlled timestamp goes backward in time is proportional to the time difference between the received timestamp and the controlled timestamp. It is shown through simulations at the end of this section that the time difference in (3.8) can be in the order of few micro-seconds at steady state. If the real timestamp is adjusted to $\overline{T}_i / s_{new}$ before the real timestamp is multiplied by s_{new}, then no backward leaps in time will be observed. However, it is necessary to wait if the difference $\left(\overline{T}_{rx\_timestamp} - \overline{T}_i\right)$ is small relative to the speed at which the computation can be made.

As a simple example of CSMNS, assume two clocks; clock 1 deviates +0.5 time units for every real-time unit (i.e., $T_1(t) = (1+0.5)t$), and clock 2 deviates −0.5 time units for every real-time unit (i.e., $T_2(t) = (1-0.5)t$), also assume zero initial conditions for both clocks. Figure 3.4a shows the behavior of the TSF assuming the faster clock (clock 1) always transmits a beacon successfully at every opportunity. Figure 3.4b uses the rule in (3.3) to update the slope of the slower clock (clock 2), that is

$$s_{new} = s_{old} + K\frac{T_{rx} - T}{T}, \tag{3.9}$$

where K is the proportional gain used to adjust the speed of convergence. T_{rx} is the timestamp embedded in the received beacon and T is the timestamp of the receiving node. s_{new} is used to obtain a new virtual time $\overline{T}_2(t) = s_{new}T_2(t)$. Figure 3.4b shows the first adjustments followed by the slower clock using $K = 0.8$ and an initial value of $s = 1$ in both clocks. Figure 3.5 is the continuation of Figure 3.4b until the 20th adjustment. Note that after the 2nd adjustment the time difference between the two clocks increases momentarily until convergence is finally achieved with better accuracy than the TSF in Figure 3.4a. Figure 3.5 also shows some of the values of the parameter *s* computed by the slower clock in every adjustment. Also observed in Figures 3.4a,b is the fact that the TSF response is oscillatory (3.4a), while the CSMNS response converges as time progresses (3.4b).

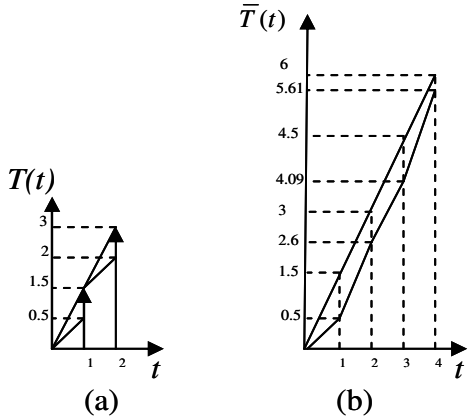

Figure 3.4 Numerical comparison between the TSF and CSMNS

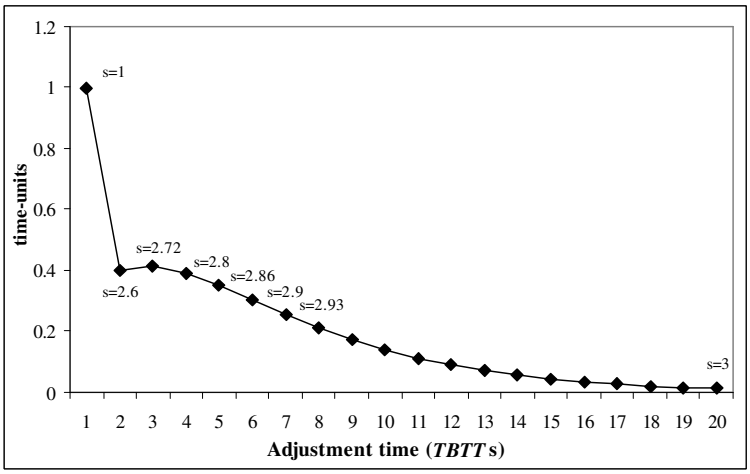

Figure 3.5 Time difference between the clocks in Figure 3.4b

The latter procedure along with Equation (3.9) forms the basis of the proposed NTS algorithm. However, the procedure in Figures 3.4b and 3.5 assume that one of the nodes always transmits a beacon successfully in every TBTT, this is obviously a simplistic assumption only meant for a didactic explanation of the idea behind the approach. This assumption is not made when obtaining the results in Figures 3.2 and 3.3.

3.2 CSMNS-Rotating Master Node

A non-hierarchical approach has the advantage that not all nodes need to contend for beacon transmission in every *TBTT*. It is sufficient if only a sub-set of all the nodes contend at any given time. In this case all nodes still have equal opportunity to transmit, but rather in a larger time-span. The following method takes advantage of this feature:

Assume each node has a counter C_i with maximum value $C_i^{\max}$. In every *TBTT*, all the nodes perform the following operations independently and in a distributed way:

1. $C_i = \max\{C_i - 1, 0\}$
2. If $C_i = 0$, contend to send the beacon following the same procedure of the IEEE 802.11 TSF, otherwise (i.e., $C_i > 0$), wait until the next *TBTT* and return to Step 1 without attempting to transmit a beacon
3. If a beacon is successfully received before the local node sends its beacon (assuming $C_i = 0$), then set $C_i = C_i^{\max}$, adjust the correction factor based on the time-stamp received, wait for the next *TBTT*, and return to Step 1
4. If a beacon is successfully received and $C_i > 0$, adjust the correction factor based on the time-stamp received and wait for next *TBTT*, return to Step 1.

Every node will contend to send its time-stamp $\overline{T}_i(t)$ embedded in a beacon if $C_i = 0$ and no beacon has been received from any other node in the present contention window. If $C_i > 0$, or $C_i = 0$ and node *i* receives a beacon before it is able to send its own, then node *i* will not contend to send its beacon in the present contention window. For a while, a group of nodes in the same locality will be listening to beacons coming from the single node, which was the winner of the previous contention. We call the latter the Rotating Master (RM) node. The RM node holds the master status temporarily until a new contention randomly replaces it by another node. In this way, all nodes have the opportunity to be a RM node for a number of *TBTT*s. CSMNS-RMN reduces the number of beacons transmitted with respect to the basic CSMNS approach and the TSF. This translates into energy savings and beacon collision reduction.

An example of CSMNS-RMN is illustrated in Table 3.1. There are 4 nodes in the same neighborhood. At *TBTT* = 1, nodes 1 and 4 contend to send their beacons

Table 3.1 CSMNS-RMN example with 4 nodes in a single-hop scenario

TBTTs / C_i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
C_1	**0**	**0**	**0**	**0**	4	3	2	1
C_2	3	2	1	**0**	**0**	4	3	2
C_3	4	3	2	1	**0**	**0**	**0**	**0**
C_4	**0**	4	3	2	1	**0**	4	3

($C_1 = C_4 = 0$), node 1 wins and becomes the RM node. Node 4 sets its counter to $C^{\max} = 5$ as soon as it receives the beacon from node 1 at *TBTT* = 1; at *TBTT* = 2, $C_4 = 5 - 1 = 4$. At *TBTT* = 4, node 2's counter reaches zero, and it contends to send its beacon against node 1. Node 2 wins the contention, but does not win RM status since node 3 also contends in the next *TBTT*. It is straightforward to see what the procedure is about afterwards. Note that on average, fewer nodes are contending in every *TBTT*. The initial values of the counters where chosen different for every node; if the values are the same, all the nodes will contend at the same *TBTT*s, and a RM node will be determined after probably some collisions, then, after $C^{\max}$ *TBTT*s all nodes will contend again to try to become the next RM node. The previous description does not take into account the wireless medium impairments, which can cause a beacon to be destroyed even when there are no collisions. CSMNS–RMN with $C_i^{\max} = 1, \forall i$ reduces to the basic CSMNS approach. Therefore, CSMNS-RMN is a more general approach than CSMNS.

3.3 Numerical Performance Evaluation of CSMNS-RMN

We performed simulations of CSMNS-RMN in Matlab. The simulations were performed under the same assumptions presented in Section 2.3 for the simulation of the TSF in a grid-topology network. Figure 3.6 shows $T_{\max}$ for 100 nodes in a single-hop network using CSMNS, with $K_p = 0.3$ and initial offsets randomly chosen in [0, 100].

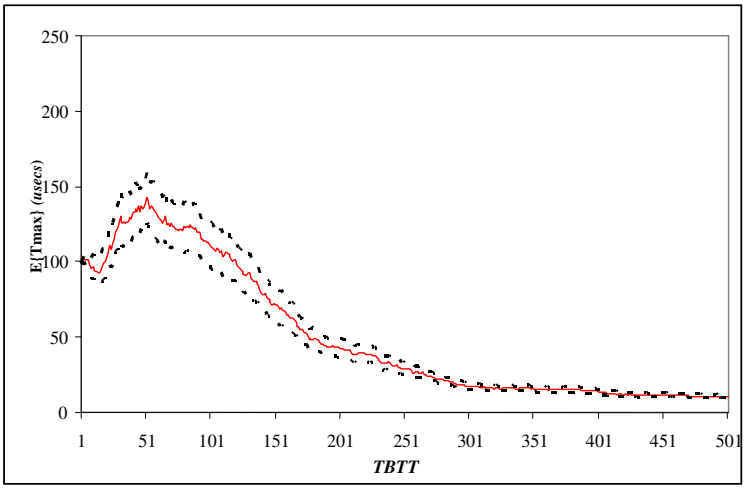

Figure 3.6 Average $T_{\max}(n)$ using CSMNS with $K_p = 0.3$, and 100 nodes in a single-hop network.

Figure 3.6 shows the ensemble average of fifty separate simulations. The dotted curves represent the 95% confidence interval over the mean estimation. Note that it approximately follows the same transient behavior shown in Figure 3.2.

Figures 3.7 and 3.8 show a sample of $T_{\max}$ in a 10x10 and a 15x15 network respectively using CSMNS-RMN. Figure 3.8 corresponds to a network that is nine times the size of the one used to obtain the TSF result in Figures 2.12 and 2.13. This shows promising scalability and accuracy performance for CSMNS. In Figure 3.7 the time of convergence defined as $t_c \equiv T_{\max}(t_c) \le 10$ $\mu secs$ is approximately 250 $secs$ using $C^{\max} = 10$ and $K_p = 0.5$, 147 $secs$ for $C^{\max} = 2$, $K_p = 0.5$, and 80 $secs$ for $C^{\max} = 2$ and $K_p = 0.8$. Decreasing $C^{\max}$ tends to improve the convergence time at the cost of increasing the overhead.

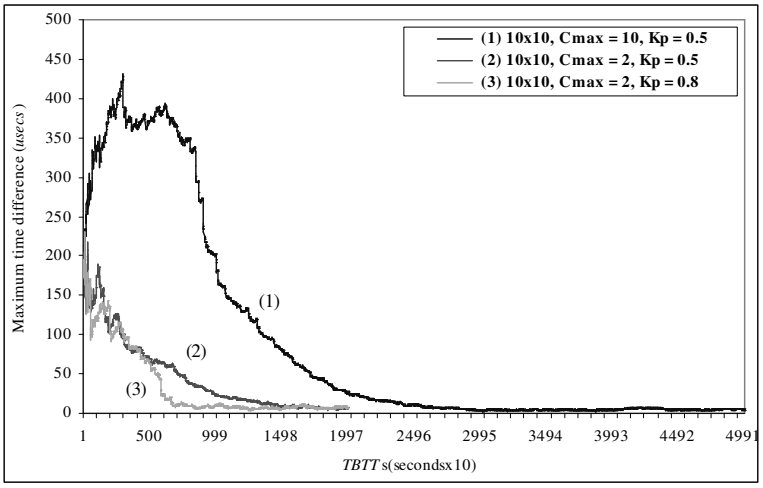

Figure 3.7 Samples of $T_{\max}$ in a 10x10 network with different $C^{\max}$ and K_p

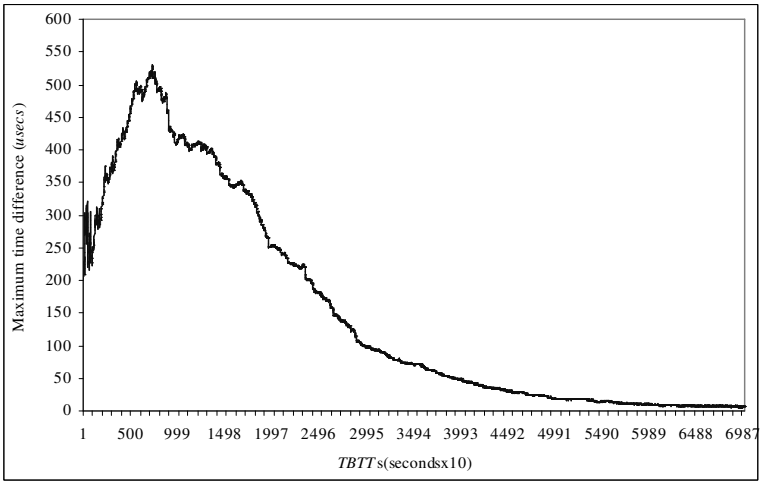

Figure 3.8 Sample of $T_{\max}$ in a 15x15 network ($C^{\max} = 10$ and $K_p = 0.5$)

An increase in the proportional gain K_p reduces the convergence time as well, but the algorithm becomes more susceptible to instability as previously discussed. The time of convergence in Figure 3.8 is 600 *secs*. Note that the clocks in the entire network start asynchronous to one another, which is a worst-case scenario. In practice, networks are

not created at once with the mutual participation of all the nodes in the network. However, even in that case CSMNS achieves synchronization. Additionally, the initial condition of Figures 3.7 and 3.8 are chosen randomly in the wider range [0, 200].

Figure 3.9 shows the ensemble average of $T_{\max}$ after performing fifty separate simulations in a 5x5 network with $C^{\max} = 10$ and $K_p = 0.5$. The dotted curves show the 95% confidence interval. As can be observed, convergence is achieved after a transition that lasts only few seconds. The final accuracy of the procedure depends on the granularity of the clock (taken as 1 $\mu secs$ in the simulations) and also in the accuracy of the estimation of delays between the PHY and the MAC layers. The mobility and wireless medium impairments in the network also affect the performance of CSMNS since the network must remain connected for the nodes to exchange their timestamps.

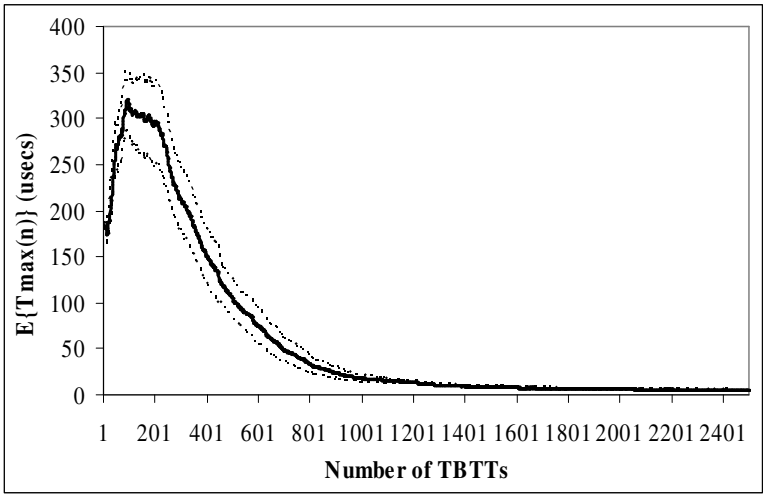

Figure 3.9 $E\{T_{\max}\}$ **in a 5x5 network using CSMNS**

Figure 3.10 shows a simulation result of a sample time difference between specific pairs of nodes when the 5x5 network is disconnected, separating the network in two equal-sized subnets. Nodes 1 and 7 are part of the first subnet, and nodes 19 and 25 are part of the second subnet (see the inset of Figure 2.12). The disconnection occurs at 200 *secs*, causing the divergence of the time differences between nodes belonging to different subnets. However the nodes belonging to the same subnet keep synchronous to one another. The network is reconnected after 200 *secs* and both subnet-times converge again.

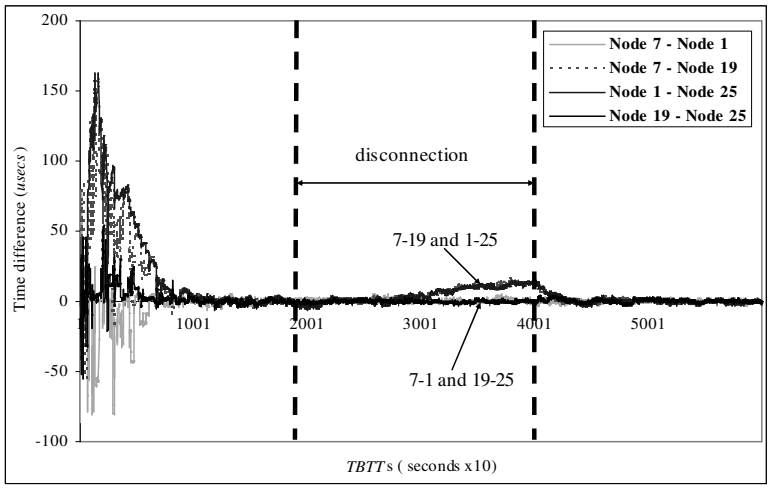

Figure 3.10 Time difference between some nodes during a network disconnection

The time synchronization in CSMNS is relative to the time skews of the clocks in the network. CSMNS does not try to synchronize with respect to UTC or any other external time reference. Figure 3.11 shows the time processes of 4 clocks in a 2x2 network. Nodes 1 and 4 deviate at +25ppm with respect to real time, and nodes 2 and 3 deviate at -25ppm. The initial time offset of all clocks is zero. After a transition period of approximately 90 *secs* all clocks are synchronous to one another. However, the common time process

deviates with respect to real time (its slope is different than zero). Figure 3.12 shows the ***s*** parameters of all the nodes corresponding to Figure 3.11. After the transition period the ***s*** parameters of all the nodes settled showing the necessary correction needed to equalize

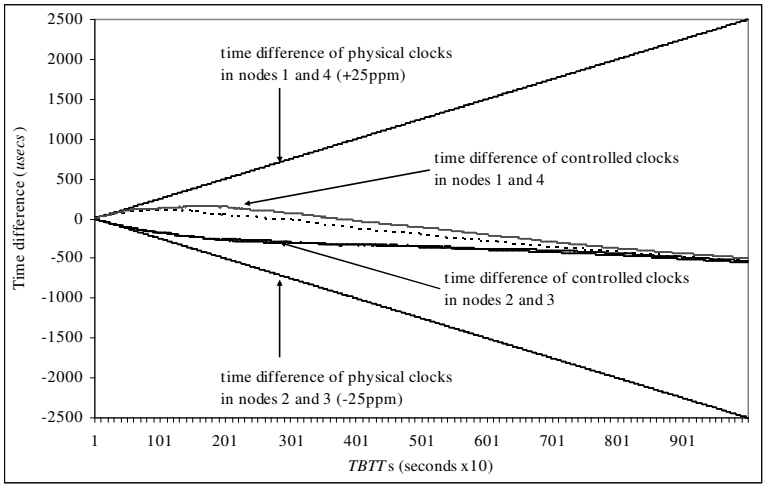

Figure 3.11 Time difference with respect to real time in a 2x2 network

($K_p = 0.01$**,** $C^{\max} = 1$**, and initial conditions = 0)**

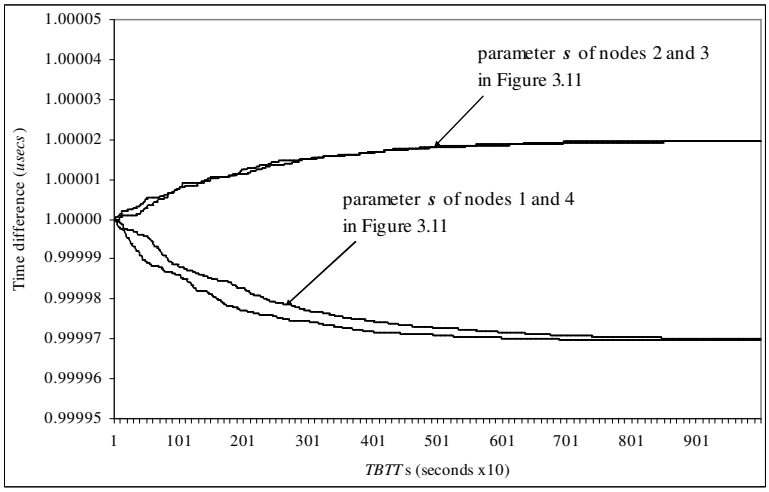

Figure 3.12 Parameter *s* corresponding to Figure 3.11

all the time processes. Nodes 2 and 3 need a larger s factor to compensate for their negative skew, and nodes 1 and 4 need a smaller s factor to compensate for their positive skew. The random way in which beacons access the medium, plus the offset of the proportional control-law with respect to the ideal, makes the system deviate from real time even when there is symmetry in the clock skews as in the previous example.

Figure 3.13 shows the final time deviation in parts per million (ppm) observed in a 2x1 network in which the clocks deviate with different skews. Each point in Figure 3.13 corresponds to a separate simulation and the time deviation observed at steady state. The ensemble average of the case in which Node 1 skews at +25ppm and Node 2 at -25ppm is -0.7 ppm, while the case in which Node 1 skews at +25ppm and Node 2 at -5ppm shows an ensemble average of 9.8 ppm. As we can see, the final time-deviation of CSMNS with respect to real-time is approximately equal to the average of the individual skews

This is an interesting fact about a mutual NTS approach. In particular, it seems possible to build a very accurate clock formed by the mutual synchronization of many less-accurate clocks. However, the latter is not the main goal of our work.

3.4 CSMNS with Permission Probabilities

In the original TSF a node will pick a slot from a contention window of $W+1$ slots in the range $[0,W]$ and try to transmit in it if the medium is idle. However, assume that even if the medium is idle a node decides to transmit based on the successful outcome of a

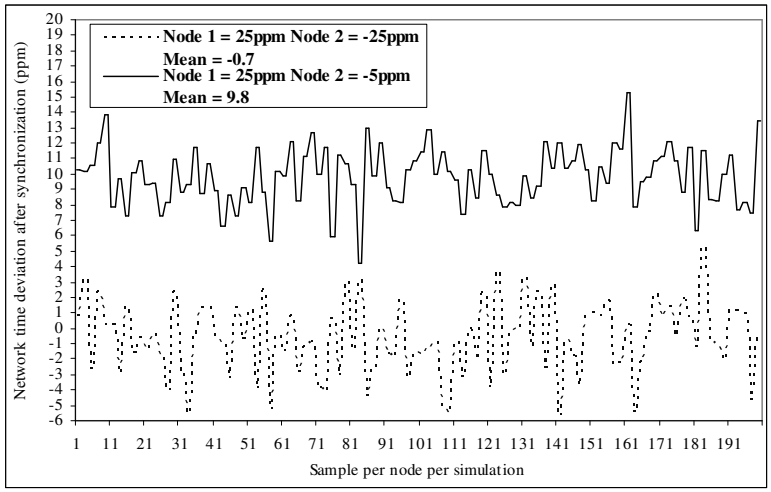

Figure 3.13 Network time-deviation with respect to real-time after synchronization is achieved in a 2x1 network

random experiment. That is, the node transmits in the chosen slot if the medium is found idle and the outcome of a random experiment (the flip of a coin) is successful. The rationale behind this extra procedure is that the beacon collision probability will be reduced if some of the nodes give up on their attempts to transmit their beacons, particularly when the number of nodes is in the order of tens or hundreds. Reducing the beacon collision probability can translate into a larger probability of having any beacon transmitted over the air (i.e., P_{any}), which in turn can reduce the convergence time of CSMNS. Note that the procedure above can enhance both the basic-CSMNS and CSMNS-RMN. Therefore, the best scheme will presumably be CSMNS-RMN with permission probabilities. In what follows we show what is the improvement attainable by using permission probabilities.

The previous procedure is incorporated into the analysis of P_{any} [17]. We extend this analysis to take into account the additional experiment described above. In what follows

$\hat{P}_{any}(n,W)$ is the probability that any beacon is transmitted successfully in a contention window of $W+1$ slots with permission probability $\overline{p}$.

The probability of the event E_1 that there is no beacon transmission in slot 0, but there is a successful beacon transmission in the remaining window (i.e., $[1,W]$) is given by the probability that i nodes $(0 \le i \le n)$ pick slot 0 and the outcome of all their random experiments are unsuccessful, and that the remaining nodes (i.e., $(n-i)$) pick a slot in the remaining window with one successful transmission. That is

$$P(E_1) = \sum_{i=0}^{n} \binom{n}{i} \left[\left(\frac{1}{W+1} \right)^i (1-\overline{p})^i \left(\frac{W}{W+1} \right)^{n-i} \right] P_{any}(n-i, W-1). \tag{3.10}$$

The probability of the event E_2 of a successful beacon transmission in slot 0 is given by the probability that j nodes $(1 \le j \le n)$ pick slot 0 while only one of them has a successful outcome from the random experiment and the rest of the $(n-j)$ nodes pick a slot in the remaining window $[1,W]$. That is

$$P(E_2) = \sum_{j=1}^{n} j \binom{n}{j} \left(\frac{1}{W+1} \right)^j \overline{p}(1-\overline{p})^{j-1} \left(\frac{W}{W+1} \right)^{n-j} \tag{3.11}$$

The probability of the event E_3 that there are collisions in slot 0, but a successful beacon transmission in the remaining window is given by the probability that i nodes $(2 \le i \le n)$ pick slot 0 and x $(2 \le x \le i)$ actually transmit, and that j $(0 \le j \le n-i)$ nodes

pick a slot in the next $(b-1)$ slots giving up their transmission attempt for the next *TBTT* due to carrier sensing, and that there is a successful transmission in the remaining window $[b,W]$. That is for $W > b$ and $n \geq 2$

$$P(E_3) = \sum_{i=2}^{n}\sum_{x=2}^{i}\sum_{j=0}^{n-i}\binom{n}{i}\binom{i}{x}\binom{n-i}{j}\left(\frac{1}{W+1}\right)^{i}\overline{p}^{x}(1-\overline{p})^{i-x}\left(\frac{b-1}{W+1}\right)^{j}\left(\frac{W-b+1}{W+1}\right)^{n-i-j}P_{any}(n-i-j,W-b) \quad . \quad (3.12)$$

Therefore $\hat{P}_{any}(n,W)$ is given by

$$\hat{P}_{any}(n,W) = P(E_1) + P(E_2) + P(E_3), \quad (3.13)$$

$\hat{P}_{any}(n,W)$ is computed for different $\overline{p}$ and different number of nodes (network sizes). The results are plotted in Figure 3.14.

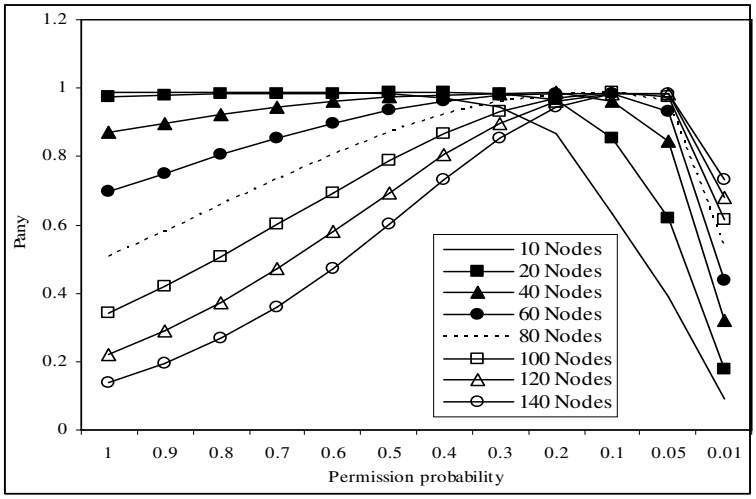

Figure 3.14 $\hat{P}_{any}(n,W)$ for different network sizes and different values of $\overline{p}$.

Note that $\hat{P}_{any}(n,W) = P_{any}(n,W)$ if $\overline{p} = 1$. $\hat{P}_{any}(n,W) > P_{any}(n,W)$ for $\overline{p} > 0.3$. The best $\overline{p}$ depends on the number of nodes and it is smaller for larger networks. The increase in the probability of transmitting any beacon is more pronounced when using permission probabilities if the number of nodes is large. When the number of nodes is small (in the order of few tens) permission probability is actually detrimental since the nodes refrain from transmitting, causing a situation referred to as beacon starvation (i.e., not enough beacons being transmitted). However, when the number of nodes is large, the action of refraining from transmitting beacons decreases the collision probability of the beacons while maintaining a good number of beacon transmissions. However, the latter is achieved if the permission probability is not too low, as shown in Figure 3.14. Figure 3.15 shows the benefit of having a permission probability $\overline{p} = 0.3$ in a network of 100 nodes when using permission probabilities over the basic-CSMNS algorithm.

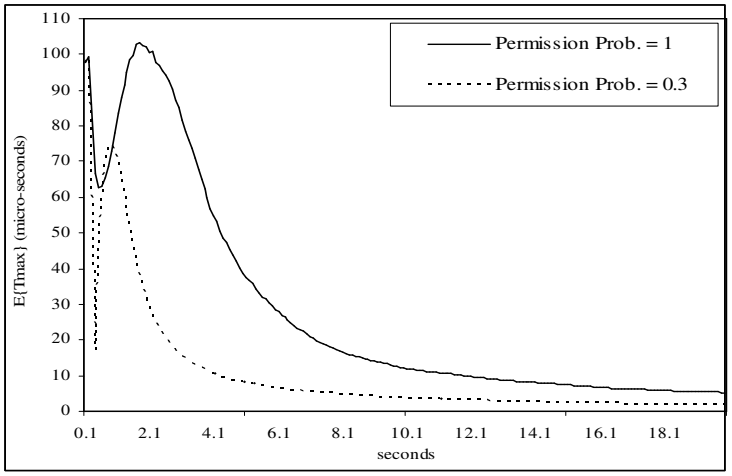

Figure 3.15 Effect of permission probability on the transient behavior of a network of 100 nodes using the basic-CSMNS algorithm ($K_p = 0.3$ and *aBeaconPeriod* = 0.1 *secs*)

The new $\hat{P}_{any}(n,W)$ is used in (3.4) to compute $E\{T_{\max}\}$ in one of the curves with FHSS parameters, $K_p = 0.3$ and *aBeaconPeriod* = 0.1 *secs*. The convergence time t_c is reduced from 12 *secs* to 4 *secs* ($t_c \equiv T_{\max}(t_c) \le 10\mu\sec s,$) and the over-shoot by 30%. It can be seen in Figure 3.14, for instance, that a value of $\overline{p} = 0.3$ is beneficial for a network between 20 and 140 nodes. However, it is detrimental for a network of approximately 10 nodes or less. Therefore, some rough estimation of the number of nodes in the network is required if permission probability is implemented for CSMNS.

3.5 Summary

A novel NTS algorithm is presented that is capable of achieving accuracies in the order of few micro-seconds in single-hop or multi-hop wireless networks with minimum infrastructure. The accuracy, however, is dependent on the estimation of PHY to MAC delays, which in most cases can be estimated with good accuracy. CSMNS is a non-hierarchical, convergent, and scalable NTS approach. One of the interesting advantages of CSMNS over previous mutual network synchronization approaches is the possibility of implementing it at the MAC layer rather than requiring special PHY layer support. This means CSMNS can bring mutual network synchronization advantages into civilian applications. Furthermore, the CSMNS approach can be integrated with a time-slotted MAC protocol if a beacon contention window is included in the frame á la IEEE 802.11. An initial beacon contention time-window can be allocated in the frame, and the rest can be utilized for synchronous time-slotted data transmissions. Finally, CSMNS is a NTS

approach specifically designed for a wireless multi-hop network or an infrastructure-less single-hop wireless network. Both of which can be comprised of mobile or static nodes. We also present extensions of the basic CSMNS approach that have the potential to increase accuracy and reduce the protocol overhead even further. Optimization in terms of the particular application of the network (e.g., sensor networks, traditional data transmission networks) can be studied as part of our future research. To the best of our knowledge CSMNS represents the first attempt to adapt the mutual network synchronization approach to the constraints and needs of civilian applications.

Chapter 4

Topology-transparent Scheduling Protocols

This chapter presents the most important results in the area of Topology-transparent scheduling protocols proposed for Wireless Ad Hoc Networks. In particular the works by Chlamtac and Farago [17], Ju and Li [39] and Syrotiuk et al., [40] are discussed as an introduction to our proposed Code-based scheduling protocols. First, an introduction and motivation are given followed by a description of the topology-transparent scheduling studies made by the authors above. Finally, a summary with some concluding remarks is presented.

4.1 Introduction

The most popular scheduling schemes for Wireless Ad Hoc networks are those who find their roots in the ALOHA protocol, such as the CSMA/CA protocol used, for instance, in the IEEE 802.11 and IEEE 802.15.4 standards. It is well known that the family of ALOHA protocols offer good performance for nodes transmitting best-effort traffic in a shared channel, however, they suffer from instability and lack of any performance guarantee due to their random nature. If real-time or priority applications are to be supported with some minimum performance guarantees it is imperative to add more structure to the scheduling approach. A possible way to add structure is by allowing the

transmissions to happen during known time boundaries or time-slots. The slotted-ALOHA protocol is one possible enhancement of the ALOHA protocol that uses this technique, however, a high degree of randomness is still inherent in the procedure and therefore no guarantees are possible. A highly structured scheduling protocol on the other hand usually suffers from being inflexible and difficult to manage in a distributed way. A traditional example of a structured scheduling protocol is classic time division multiple access (TDMA). In classic TDMA every node is assigned, *a-priori,* a unique slot (channel) in which to transmit, in this way collisions are avoided and the packet delay becomes highly deterministic at the data link layer level. However, the throughput and delay efficiency of such a scheme is questionable due to the lack of a time-slot re-utilization technique that takes advantage of the spatial separation between the nodes. The need for network time-synchronization in a wireless Ad Hoc network is a non-trivial factor that also needs to be taken into account when implementing such structured scheduling approaches. Furthermore, additional signal overhead might be needed to ensure the unique time-slot assignment per node. Other, more elaborated, TDMA schemes have been proposed. However, the overhead cost to cope with topology changes is usually non-negligible.

More recently, an approach referred to as topology-transparent scheduling was proposed in [17] for a multi-hop packet radio network based on polynomials over a Galois field. Each user is assigned a different polynomial with degree k and coefficients in GF(q), which it uses to select a slot in a frame. The difference between two polynomials also results in a polynomial of degree less than or equal to k, therefore the number of roots of the difference between two polynomials will be bounded by k as well.

This translates into the fact that the maximum number of common points between any two different polynomials will be *k*. Additionally, if one has information on the maximum number of interferers that any node could have in the network at any given time, it could be possible to guarantee a minimum of performance as long as the scheduling is performed following the unique polynomial evaluation. The latter concepts will be made clearer in the upcoming sections.

The scheduling of channels using polynomials in a Galois field seems to have been originated in the early works in [41] and more notably [42]. Solomon [42] proposed an optimal solution for the *hit* problem in a frequency hopping multiple access (FHMA) system that resembles the construction and arguments made in [17]. A *hit* is nothing more than a collision between two or more transmitters at a common receiver caused by the use of the same channel. Solomon's idea was to use maximum distance separable (MDS) sequences over a Galois field in order to minimize the *hit* probability. A well-known set of sequences with MDS properties can be constructed by a generalization of the method used in [17] and constitute the code-words of a Reed-Solomon (RS) code. In fact, the scheduling procedure designed in [17] corresponds to the code-words of a singly extended RS code. RS codes can be extended, augmented, truncated, and shortened. Therefore, the proposed procedure in [17] can be seen as a specific case of a more general scheduling approach based on RS codes.

A more recent work [40] identifies a generalization to the procedure in [17] based on Orthogonal Arrays (OAs). A large number of different code constructions produce OAs, including the RS, BCH, and Reed Muller code constructions among many others. We argue that, in principle, any linear or non-linear *q-ary* code can be used as a scheduling

sequence. Additionally, all the possible modifications a code can undergo (e.g., truncation, augmentation, extension, shortening) represent an obvious generalization over the same code construction. It is important at this stage, however, to find code constructions and techniques that can offer improved performance and simpler implementation of the topology-transparent scheduling protocols based on codes. A way to achieve the latter is the subject of the next chapter.

4.2 Topology-transparent Scheduling Protocols

Chlamtac and Farago [17] were the first to propose a topology-transparent scheduling algorithm based on the evaluation of polynomials over a Galois field in the context of multi-hop packet radio networks. They state that their algorithm can be used for any multiple channel access protocol (e.g., time, code or frequency-division multiple access). However, for brevity they refer to the TDMA case. The network is viewed as an undirected graph *G(V,E)* with *N* nodes, where *V* is the set of nodes ($|V| = N$), and *E* is the set of edges or links. The degree of a node v ($\deg(v)$) is the number of neighbors of node v, and the maximum degree $D_{\max}$ is the global maximum degree in the network $\left(\text{i.e.}, \max_{v \in V} \deg(v) \right)$. Time is slotted, and a frame is comprised of *L* slots. A frame *F* is seen as a set of slots: $F = \{s_0, s_1, ..., s_{L-1}\}$. The scheduling assignment of node *v* is then given by a set $S_v \in F$, where S_v is the set of slots in which node *v* can transmit.

The objectives of their algorithm are as follows:

1. For each node *v*, each neighbor *u* of *v*, and each neighbor $u' \neq v$ of *u*, there should be at least one slot $s \in S_v$ such that $s \notin S_u$ and $s \notin S_{u'}$. They claim this will allow node *v* to transmit without collisions at least once in every frame. This requirement is equivalent to state that at least once in a frame a node will use a slot that is different to the ones used by its intended receiver (to avoid primary collision) and the neighbors of its intended receiver (to avoid secondary collisions)
2. The slot assignment depends on global parameters *N* and $D_{\max}$ only
3. The frame length *L* should be significantly smaller than *N* (i.e., *N* would be the frame size used by a classic-TDMA scheduling approach).

Recall that the destruction of a packet in a given receiver does not necessarily depend only on first and second-hop neighbor transmissions since the transmission and interference ranges are usually different in real wireless networks. Furthermore, the values of *N* and $D_{\max}$ could prove to be difficult to obtain in a highly dynamic wireless environment. However, it is assumed that an upper bound to the actual values is either known or enforced. The enforcement of a given degree in a wireless network is beyond the scope of our work, and also the work in [17].

The description of their algorithm follows (for a good exposition of finite field theory see [43]). Let GF(*q*) be a Galois field of order *q*. Let $q = p^m$, *p* a prime and $m \geq 1 \in \mathbf{Z}^+$ is an arbitrary positive integer. Every element in GF(*q*) is labeled with the integers 0, 1, …, *q*-1. Assign every node *v* in the network a unique (otherwise arbitrary) vector-identifier polynomial $VID_v[x]$ of degree *k* with coefficients in GF(*q*). That is

$VID_v[x] = a_k x^k + a_{k-1} x^{k-1} + \cdots + a_0$, where $a_j \in GF(q)$, for $j = \{k, k-1, ..., 0\}$.

Define a frame of size $L = q^2$ slots. Determine the set of slots $S_v \in F$ for every node in the network as follows:

for every $v \in V$

$S_v(0 : q-1) = \mathbf{0}$; //initialize the values of the *q* slots in which *v* is going to transmit

for $(i = 0 : q-1)$ // for every value in the set = {i = 0,1,…,*q*-1)

$S_v(i) = iq + VID_v[i]$; //obtain each slot value by evaluating $VID_v[x]$

end

end

In the previous algorithm, each unique polynomial is evaluated for every value between 0 and *q*-1 and assigned to the node's slot assignment set *S*. Note that an alternative way to visualize the frame is as composed of *q* sub-frames with *q* slots each, as shown in Figure 4.1. In this case node *v* will evaluate its unique polynomial with the sub-frame value in order to obtain the slot number assigned to it in every sub-frame. Recall that network time-synchronization is assumed. Requirements 1 and 2 above are satisfied by constraining the values of *q* and *k* as follows

$$q^{k+1} \geq N, \tag{4.1}$$

$$q \geq kD_{\max} + 1. \tag{4.2}$$

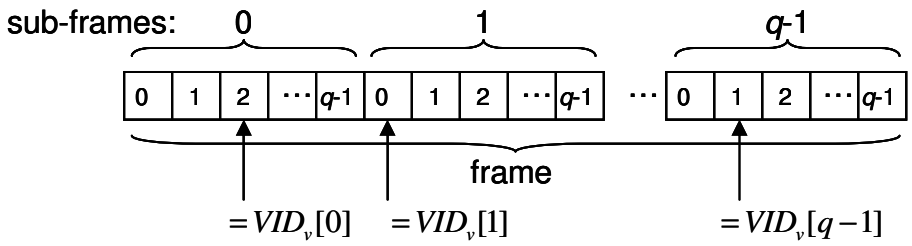

Figure 4.1 Frame structure of the algorithm proposed by Chlamtac and Farago

Inequality (4.1) guarantees that every node in the network will have a unique *VID*[x] polynomial, therefore excluding the possibility of the topology-dependent problem of assigning *non-interfering* polynomials to the nodes in the network. Inequality (4.2) ensures requirement 1 above is satisfied assuming only up to two-hop neighbor interference. That is, the nodes that can interfere with a transmission from node *v* to node *u* are assumed to be the neighbors of node *u* except *v*, and node *u* itself. This is equal, in the worst case, to the maximum degree of the graph modeling the network (see Figure 4.2). Any two different polynomials $VID_v[x] \neq VID_u[x]$ of degree *k* will coincide in the

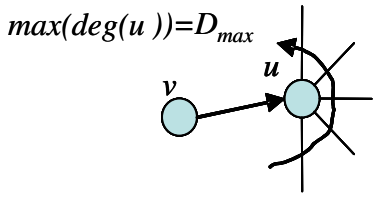

Figure 4.2 Assumed maximum number of interferers in Chlamtac's algorithm

same value after being evaluated at a common point p iff p is a root of the polynomial obtained by the difference between both polynomials (i.e., $VID_v[p] - VID_u[p] = 0$). The difference between two k^{th} degree polynomials is another polynomial in the same field with degree $\leq k$, therefore the maximum number of points in which two polynomials coincide, and hence the maximum number of slots in which the two nodes collide is given by k.

Combining the two previous conclusions we can state that the maximum number of collisions a node can perceive when transmitting to another node is kD_{max}. It is apparent then, that in order to guarantee at least one transmission without collision in every frame we have to ensure that q and k satisfy the inequalities (4.1) and (4.2).

The following algorithm is proposed in [17] in order to find the minimum q subject to constrains (4.1) and (4.2). That is, the minimum frame size subject to having at least one successful transmission per frame.

Assume $q_1, q_2 \ldots$ to be the sequence of increasing powers of primes (i.e., 2, 3, 4, 5, 7, 8, 9…), then

$i = 0;$

do

$$i = i + 1;\ q = q_i;\ k = \lfloor (q-1)/D_{\max} \rfloor;$$

until ($k \geq 1$ and $q^{k+1} \geq N$)

For instance, if there are N = 100 nodes, and $D_{\max} = 4$, the algorithm above will find k = 2 and q = 9, which represents a frame size of $L = q^2 = 81 \text{slots} (< 100)$ slots, and sub-frame size of 9 slots. In the previous example, any two nodes will collide at most once in every frame.

Ju and Li [39] proposed an optimal topology-transparent scheduling approach based on [17] that optimizes an expression representing the minimum throughput of any node in a network. The procedure is the same as in [17] and depicted in Figure 4.1. The novelty in their work lies in the fact that they found a way to compute values of q and k that maximize "minimum throughput" rather than focusing on the minimization of the frame size. In this way they generally obtain larger minimum throughput and smaller maximum delay than the method proposed in [17]. In particular, recall that $kD_{\max}$ is the maximum number of collisions a node can suffer in a frame, and the number of transmission attempts of a node in a frame is q. The authors define minimum throughput ($G_{\min}$) as

$$G_{\min} = \frac{q - kD_{\max}}{q^2}. \tag{4.3}$$

Recall that the size of the frame is $L = q^2$. That is

$$G_{min} = \frac{Number\ of\ trans.opportunities - (Max.\ pairwise\ number\ of\ collisions)\cdot(Max.degree)}{Frame\ size} \tag{4.4}$$

They found the maximal $G_{\min}$ for a given value of k following a classical optimization procedure over the field of real numbers. The latter is technically an approximation due to the fact that the design variables (q and k) admit only integer values. Their result is summarized as follows

$$\max(G_{\min}) = \begin{cases} \dfrac{1}{4kD_{\max}} & if \;\; N^{1/(k+1)} \le 2kD_{\max} \\ \dfrac{\left(N^{1/(k+1)} - kD_{\max}\right)}{N^{2/(k+1)}} & otherwise \end{cases} \tag{4.5}$$

Their algorithm first computes the value of q, given k and the computed $G_{\min}$ (4.5), and proceeds in the same fashion as the algorithm in [17] afterwards. Their algorithm consistently shows larger values of $G_{\min}$ and smaller values of maximum delay than [17]. Therefore the authors in [39] argue their scheme can guarantee a minimum throughput that is larger than Chlamtac's approach. However, the guarantee must be understood within the assumptions made (interference up to second-hop neighborhoods, and knowledge of $D_{\max}$ and N, which were the assumptions made in [17] as well). Ju's approach has the additional advantage that it has smaller delay variability (i.e., delay-jitter), which is attractive for real-time applications. To see this, define the following lower and upper bounds for the delays [39]:

$$\begin{aligned} DT_{\max} &= (\text{frame length})/(\text{minimum number of successful transmissions in a frame}) \\ &= q^2/(q - kD) = 1/G_{\min} \end{aligned},$$

$$DT_{\min} = (\text{frame length})/(\text{maximum number of successful transmissions in a frame})) = 1/G_{\max}$$

Figure 4.3 qualitatively shows the relative positions of $G_{\min}$, $G_{\max}$, $DT_{\min}$, and $DT_{\max}$ based on the results in [39]. Note that the difference between maximum and minimum values of delay is narrower for Ju's approach. Therefore the delay variability will be constrained in a narrower region. The apparent optimality of Ju's algorithm will be shown to be valid only within a given construction and not in a more global sense since, for instance, RS codes always exist that are longer than the given order of the code, therefore frames of more than q sub-frames are always possible with the longer code keeping the MDS property (i.e., $d_{\min} = n - k + 1$). The work in [40], plus additional work in [44] and [45], identify the polynomial construction in [17] as one of the possible constructions used for orthogonal arrays (OA). A large number of different code constructions produce OAs and others do not [46]. Different constructions may have different performance if analyzed separately as shown in Chapter 5. Therefore, a different view of the problem is required.

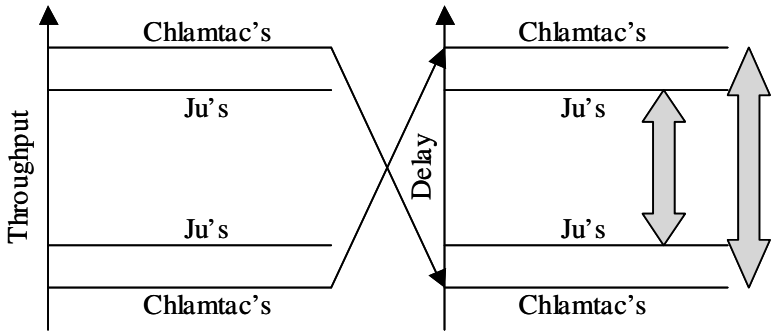

Figure 4.3 Delay jitter expected in Ju's and Chlamtac's approach

4.3 Summary

This chapter presented the most relevant studies and ideas in the area of topology-transparent scheduling protocols. It represents an introduction and motivation to our proposed code-based scheduling protocols. We finalize this chapter concluding that previous work in this area has not been general enough, and has not provided constructive ways to approach the scheduling problem in wireless Ad Hoc network. In the context of Wireless Ad Hoc networks, all previous work has been tied to the original polynomial construction proposed by Chlamtac and Farago [17], including the generalization proposed in [40] based on OAs. The main point is that the construction proposed in [17] is not the only possible construction. A more general and constructive approach is presented in the next chapter that makes use of the large number of code constructions available from the field of coding theory.

Chapter 5

Code-base Scheduling Protocols for Wireless Ad Hoc Networks

This chapter presents a generalization to the topology-transparent scheduling protocols originally proposed in [17] in the context of multi-hop packet radio networks. We re-introduce the topology-transparent scheduling problem using the terminology of coding theory and refer to it as code-based scheduling. We first present our approach and compare the throughput and delay performance of scheduling schemes based on two code constructions that are known to have some desirable properties, the Reed-Solomon (RS) codes, which are maximum distance separable, and the Hermitian codes, which are longer than RS codes for a given alphabet size and posses a good minimum relative distance ($d_{\min} / length$) which, as explained in this chapter, is a key metric for the realization of better minimum throughput guarantees. We compare the average throughput and delay performance between code-based and contention-based protocols represented by a code-based scheduling protocol that uses Orthogonal Arrays and the slotted-ALOHA protocol respectively. Finally, we investigate the effects of code-word selection on the average performance of a code-based scheduling approach, and present a hybrid scheduling protocol that combines contention-based and code-based scheduling.

5.1 Introduction

A code-based scheduling protocol utilizes the different code-words of a structured code to select the slots over which the nodes in the network will transmit. A structured code is constructed using mathematical techniques aimed at creating a set of code-words with some desirable properties. We exclude in this definition the use of code-words formed by elements drawn from a finite and discrete random distribution. Code-based scheduling protocols are a subset of the topology-transparent scheduling protocols. Note that some protocols that do not use structured code-words can also be topology-transparent (e.g., slotted-ALOHA), therefore we prefer the term code-based scheduling. The transparency of a scheduling protocol implies that the procedure does not depend, and its performance does not get affected, by the relative position of the nodes in the network. However, some dependency on general network parameters, such as the number of nodes or the maximum density of the network, is generally accepted. Examples of structured codes are numerous and a possible way to classify them is based on their method of construction. Codes constructed using algebraic principles include the popular Reed-Solomon (RS) codes and the Hermitian codes; whereas codes constructed with more empirical methods include the convolutional and turbo codes.

A code is denoted as $C(n, k, q)$ where n is the length of the code-words, k is the rank of the code, and q is the order of the Galois field over which the code is defined [47]. A $C(n, k, q)$ code has q^k code-words of length n in GF (q). The code-words of *any* code can be used to schedule channel-slots (e.g., codes, time or frequency slots) in a multiple access system.

5.2 Problem Model and Comparative Evaluation of Reed-Solomon and Hermitian codes

Assume a code $C(n, k, q)$ of length n, rank k in GF(q). C(n, k, q) has q^k code-words of length n in an alphabet formed by the elements of GF(q). The code-words of such code can be utilized as scheduling patterns for the nodes of a wireless Ad Hoc network as suggested in the previous section. Every node accesses the medium assuming a time-slotted structure as shown in Figure 5.1. The $C_i(x)$ in Figure 5.1 represents the x^{th} digit of the code-word assigned to node i used to schedule the transmissions in the x^{th} sub-frame.

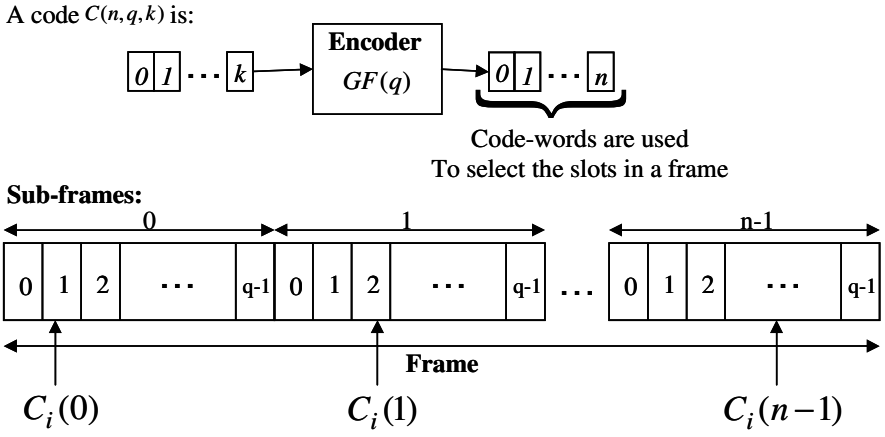

Figure 5.1 General time-slotted structure of a code-based scheduling protocol

A lower-bound throughput of a node in a network with code-based scheduling can be written as

$$G_{\min} = \frac{n-(n-d_{\min})I_{\max}}{nq}. \tag{5.1}$$

Where $I_{\max}$ is the estimated maximum number of interferers a node can have at any given time, $d_{\min}$ is the minimum Hamming distance of the code, q is the order of the finite field in which the code is defined, and n is the length of the code. The maximum degree $D_{\max}$, used in previous works (e.g., [17], [39]), has been substituted for the more appropriate $I_{\max}$, in this way nodes more than two-hops away could be considered when modeling the interferers of a given node (this is a more realistic situation in a wireless medium). An $I_{\max}$ bound could be achieved with techniques such as power control or any other topology control mechanism. However, this is a topic beyond the scope of this work. The numerator in (5.1) is the minimum possible number of successful slots in a frame of nq slots. $G_{\min}$ is a lower bound since the code-words of any code will be separated by a Hamming distance larger than or equal to $d_{\min}$.

The following inequalities are imposed on the parameters of a code in order to guarantee a positive value of G_{min} in a network

$$\begin{aligned} &q^k \geq N \\ &n-(n-d_{\min})I_{\max} > 0 \Rightarrow n \leq \frac{1-d_{\min}I_{\max}}{1-I_{\max}}. \\ &q \geq I_{\max}+1 \end{aligned} \tag{5.2}$$

N is the number of nodes in the network. The first inequality in (5.2) guarantees a unique code-word for every node in the network, and the second inequality ensures that a node

will have at least some successful transmissions (greater than zero) in a frame (see equation 5.1). That is, a node should be assigned a number of opportunities to transmit in a frame greater than the maximum number of collisions possible (i.e., $n > (n - d_{\min})I_{\max}$) if one wishes to have a minimum of performance greater than zero. The third inequality is also necessary since the size of a sub-frame needs to be larger than the number of local interferers in order to be able to guarantee a minimum of collision-free transmissions.

One important design goal is to find the order and rank of the specific code for which (5.1) is maximized constrained to (5.2). It is necessary however, to know how to compute the minimum Hamming distance and length of the code. Additionally, the parameters $I_{\max}$ and N of the network must also be available (i.e., at least upper bounds to these parameters are necessary if a minimum throughput is to be guaranteed). Equation (5.1) can be re-written as

$$G_{\min} = \frac{1}{q} - \left(1 - \frac{d_{\min}}{n}\right)\frac{I_{\max}}{q}. \tag{5.3}$$

We note that code constructions that produce codes with larger $d_{\min}/n$ ratio for a given order q and $I_{\max}$ will have larger $G_{\min}$ guarantees. In other words, larger $G_{\min}$ are possible for codes with larger *relative minimum distance*. This is the reason an extended RS code will guarantee a larger minimum throughput as will be proved shortly. Hermitian codes are an example of long codes (i.e., longer than RS codes) that possess a good $d_{\min}/n$ ratio. Hermitian codes are constructed using algebraic geometry (AG) principles. In particular, Hermitian codes are derived from a Hermitian curve in a finite

field. In fact, RS codes can be seen as AG codes over a straight line in a finite field, and therefore form a specific case of the more general set of AG codes. AG codes derived from many algebraic curves can be constructed, including Elliptic and Hyper-elliptic curves. However, Hermitian curves have more points per given order; this is one of the reasons that, for some code parameters, the Hermitian codes posses a larger relative minimum distance than the RS codes of the same order. First, however, we prove that a doubly extended RS code offers larger minimum throughput guarantee than a non-extended or a singly extended RS code and compute the optimal value of q that maximizes (5.1).

An RS code can always be doubly extended without loosing its maximum distance separable (MDS) property [47] (the method of constructing this extension can be found, for instance, in [47]). A triple extension of an RS code is also possible without loosing the MDS property, however, only when the code has the following parameters $(n,k,q)=(2^m+2,3,2^m)$ or $(2^m+2,2^m-1,4)$ [47]. The minimum distance of a RS code is given by $d_{\min}=n-k+1$, substituting the latter in (5.1) yields

$$G_{\min}^{RS}=\frac{n-(k-1)I_{\max}}{nq}. \tag{5.4}$$

Non-extended, singly-extended, and doubly extended RS codes have $n=q-1,\ q,$ and $q+1$ respectively. Substituting the latter values of n in (5.4) we obtain the minimum throughput for each code version,

$$G_{\min}^{neRS} = \frac{q-1-(k-1)I_{\max}}{(q-1)q}$$
$$G_{\min}^{seRS} = \frac{q-(k-1)I_{\max}}{q^2} \quad . \tag{5.5}$$
$$G_{\min}^{deRS} = \frac{q+1-(k-1)I_{\max}}{(q+1)q}$$

We take the ratios of the minimum throughputs in (5.5) and expand the factors to get

$$\frac{G_{\min}^{seRS}}{G_{\min}^{deRS}} = \frac{q^2+(1-KI_{\max})q-KI_{\max}}{q^2+(1-KI_{\max})q} < 1 \Rightarrow G_{\min}^{seRS} < G_{\min}^{deRS}$$
$$\frac{G_{\min}^{neRS}}{G_{\min}^{seRS}} = \frac{q^2-(1+KI_{\max})q}{q^2-(1+KI_{\max})q+KI_{\max}} < 1 \Rightarrow G_{\min}^{neRS} < G_{\min}^{seRS} \quad . \tag{5.6}$$

Where $K = k-1$. Therefore $G_{\min}^{deRS} > G_{\min}^{seRS} > G_{\min}^{neRS}$. The latter implies the fact that the results found in [39] are not optimal since larger $G_{\min}$ are possible with the same values of k and q. Note that k in this and the following sections represents the rank of the code as it is frequently used in the coding theory literature. This is different to the k used in [39], which, in that case, represents the maximum degree of the polynomial used to construct the codes. The relationship between both is $k_{[24]} = k-1$. Where $k_{[24]}$ is the k used in [39].

The value of q that maximizes $G_{\min}^{deRS}$ can be found by solving the following

$$\frac{\partial}{\partial q}\left(\frac{q+1-KI_{\max}}{(q+1)q}\right) = 0 \Rightarrow q^* = KI_{\max} - 1 \pm \sqrt{KI_{\max}(KI_{\max}-1)} \, . \tag{5.7}$$

In order to satisfy the second inequality in (5.2), namely $n \geq (n - d_{\min})I_{\max} + 1 \Rightarrow$ $q + 1 \geq KI_{\max} + 1$, we must choose

$$q^* = KI_{\max} - 1 + \sqrt{KI_{\max}(KI_{\max} - 1)}, \tag{5.8}$$

which after substitution in $G_{\min}^{deRS}$ results in

$$\max\{G_{DE\min}\} = \begin{cases} \dfrac{\sqrt{KI_{\max}(KI_{\max}-1)}}{\left(KI_{\max} + \sqrt{KI_{\max}(KI_{\max}-1)}\right)\left(KI_{\max} - 1 + \sqrt{KI_{\max}(KI_{\max}-1)}\right)} & for\ q \geq KI_{\max} \\ \dfrac{N^{1/(K+2)} + 1 - KI_{\max}}{\left(N^{1/(K+2)} + 1\right)N^{1/(K+2)}} & otherwise\ \ (q = N^{1/K+2}) \end{cases}. \tag{5.9}$$

Figure 5.2 plots $\max\{G_{DE\min}\}$ versus the result obtained in [39] for the case when $q \geq KI_{\max}$ and assuming $D_{\max} = I_{\max}$ to make the comparison meaningful. Note that a doubly extended RS code outperforms the proposed scheme in [39], which is actually the performance of an optimal singly extended RS code. However, the difference becomes negligible for $KI_{\max} > 10$. That is, as the node density increases in the network, the difference between both codes becomes negligible.

Next, we compare the RS and Hermitian code constructions in terms of minimum throughput and maximum and minimum delay. First, we provide a brief description of the Hermitian codes.

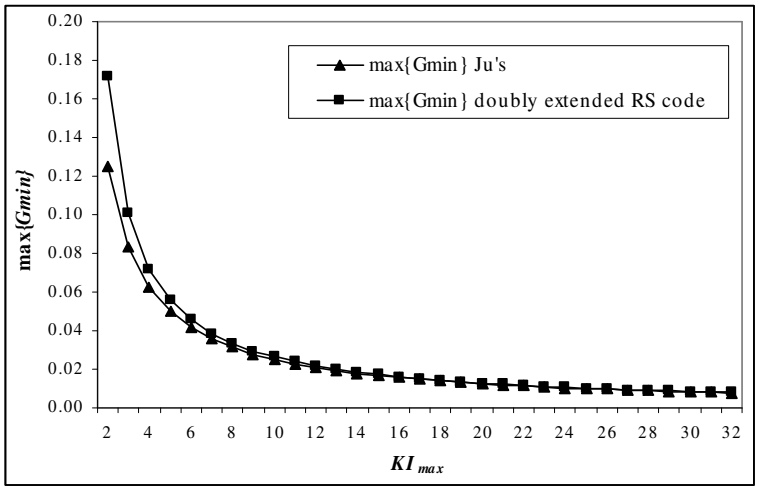

Figure 5.2 max{G_{min}} for Ju's result and for a doubly extended RS code

Definition 3.1: Let q be a power of some prime p. The Hermitian curve over $GF(q^2)$ is given by the following equation

$$x^{q+1} = y^q + y. \tag{5.10}$$

The number of points (x, y) satisfying (5.10) over $GF(q^2)$ are $n = q^3$ not counting the point at infinity [48]. Goppa suggested the use of this kind of algebraic curves for the construction of long codes [49].

Construction of Hermitian codes: Let the points on a Hermitian curve over $GF(q^2)$, except for the point at infinity, be $\{P_1, P_2, \cdots, P_n\}$. Given an arbitrary positive integer u, construct a set of functions denoted $L(uP_\infty)$ as follows

$$L(uP_\infty) = span\{x^r y^s \mid rq + s(q+1) \le u\}, \tag{5.11}$$

where $span\{\}$ encloses the set of functions that span a given space.

The Hermitian code-words of $H(q^3, k, q^2)$ can be obtained by evaluating all the functions $f \in L(uP_\infty)$ at the points $\{P_1, P_2, \cdots, P_n\}$, where k is the total number of functions f found using (5.11). The main reason why we use Hermitian codes is because they are relatively long codes with good minimum distance properties, which suggests a larger minimum relative distance than the RS codes.

Note that the previous construction is very similar to the construction of an RS(n, k, q) code if $L(uP_\infty)$ is substituted by $P = span\{1, x, x^2, \cdots, x^k\}$, which is nothing more than all polynomials of degree less than or equal to k. Every such polynomial is evaluated over the points on a straight line in $GF(q)$ (i.e., all the different elements in $GF(q)$).

The minimum distance of a Hermitian code satisfies, in general, the following inequality

$$d_{\min}^{H} \geq q^3 - u\,. \tag{5.12}$$

For the following comparisons we find the Hermitian and RS codes that maximize the minimum throughput in (5.1) subject to (5.2). Note that the only orders allowed in a Hermitian code are of the form q^2, where q is the power of some prime. The minimum distance of the Hermitian code, necessary for the evaluation of (5.1), is computed according to the following expression

$$d_{\min}^{H} = \min_{H_{cw} \in H,\, H_{cw} \neq \mathbf{0}} weight\{H_{cw}\}\,.$$

Where H_{cw} is an arbitrary codeword different to the all zero code-word. Minimum and maximum delay (i.e., $DT_{\min}$, $DT_{\max}$) are defined in Section 4.2. For instance, for a doubly extended RS code $DT_{\min}$, and $DT_{\max}$ take the following form for any value of q

$$\begin{aligned} DT_{\max} &= (q+1)q/\left(q+1-(k-1)I_{\max}\right) \\ DT_{\min} &= q \end{aligned}.$$

Figures 5.3 and 5.4 show $\max\{G_{\min}\}$ and $DT_{\min}$, $DT_{\max}$ for N = 100 nodes. Note that both constructions have roughly the same performance, except at lower densities (or smaller number of maximum interferers), in which the doubly extended RS-code (DE-RS) has a better performance in terms of minimum throughput than the Hermitian code.
In classic TDMA each node has a unique slot assigned to it out of the N slots in a frame. For instance, for a network with 100 nodes, each node has a unique slot in a frame of size 100 slots. Therefore, the throughput of a node is guarantee to be *1/N* and the delay is equal to N slots, this is shown in Figures 5.3 and 5.4. Note that the performance of code-based scheduling approaches falls below classic TDMA as the number of possible interferers increases beyond a certain threshold. In general, the performance of code-based approaches will degrade with respect to classic TDMA as the node density increases (taking the reasonable assumption that node density will increase the number of interferers a node's receiver can have). In the limit, when all the nodes are one-hop from one another (e.g., an access point or base station serving some nodes, or a single-hop wireless local area network) the classic TDMA approach will have better than or equal performance than any code-based approach.

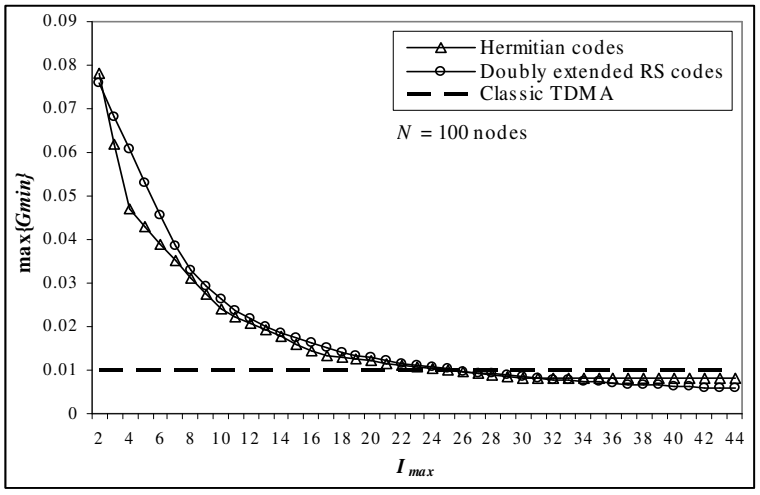

Figure 5.3 max{G_{min}} using DE-RS and Hermitian codes in a network of 100 nodes

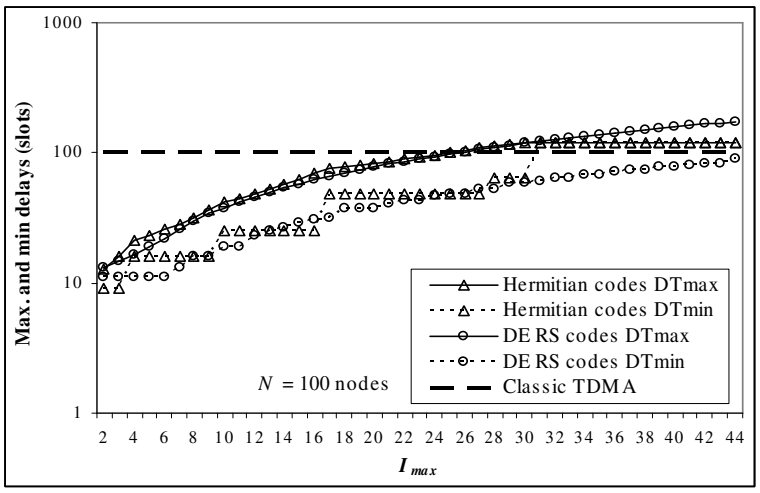

Figure 5.4 Maximum and minimum delays (in slots) using DE-RS and Hermitian codes in a network of 100 nodes

To see this take the third inequality in (5.2) and substitute $I_{\max} = N - 1$ (the maximum number of interferers in a single-hop network of *N* nodes) then

$$q \geq N . \tag{5.13}$$

The minimum throughput under this condition is

$$G_{\min}^{s-hop} = \frac{n-(n-d_{\min})(N-1)}{nq}. \tag{5.14}$$

Equation (5.14) is maximized for a given n, $d_{\min}$, and N when q attains the smallest value constrained to (5.13), that is when $q = N$, then re-arranging (5.14) after substituting $q = N$ yields

$$\max\{G_{\min}^{s-hop}\} = \frac{1}{N} - \frac{(n-d_{\min})(N-1)}{nN}. \tag{5.15}$$

Comparing (5.15) with the throughput of classic TDMA (i.e., 1/N) it is clear that since $n \geq d_{\min}$ in any code, we have $\max\{G_{\min}^{s-hop}\} \leq G_{TDMA}$.

Figures 5.5 through 5.10 show the $\max\{G_{\min}\}$ $DT_{\min}$, and $DT_{\max}$ for N = 500, 2000, and 10000 nodes. As the number of nodes increases, the Hermitian code construction offers higher minimum throughput guarantees than the RS code, particularly at lower number of maximum interferers. As the number of nodes increases, the $I_{\max}$ threshold below which Hermitian codes offer better performance than RS codes increases. The minimum and maximum delay of Hermitian codes tend to be less than the corresponding ones for RS codes as the number of nodes increases as well. The previous characteristic makes Hermitian codes attractive for large cooperative sensor networks in which

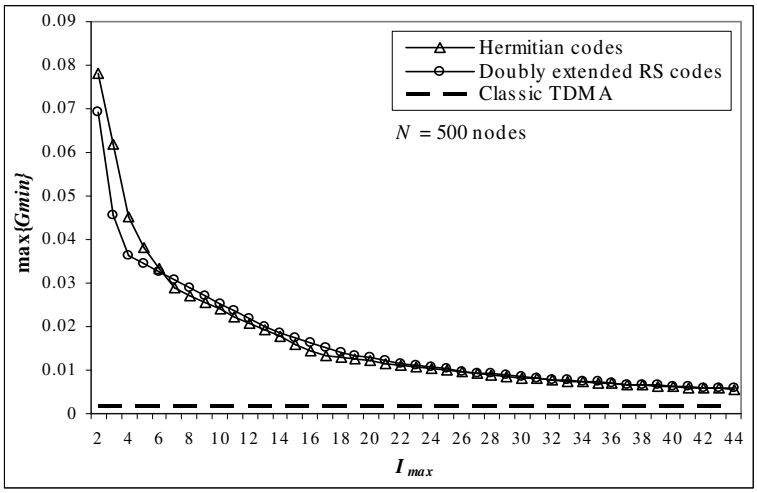

Figure 5.5 max{G_{min}} using DE-RS and Hermitian codes in a network of 500 nodes

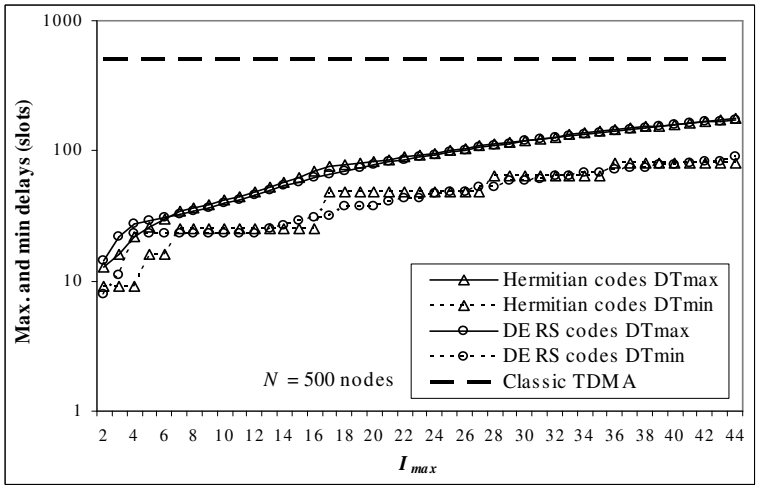

Figure 5.6 Maximum and minimum delays (in slots) using DE-RS and Hermitian codes in a network of 500 nodes

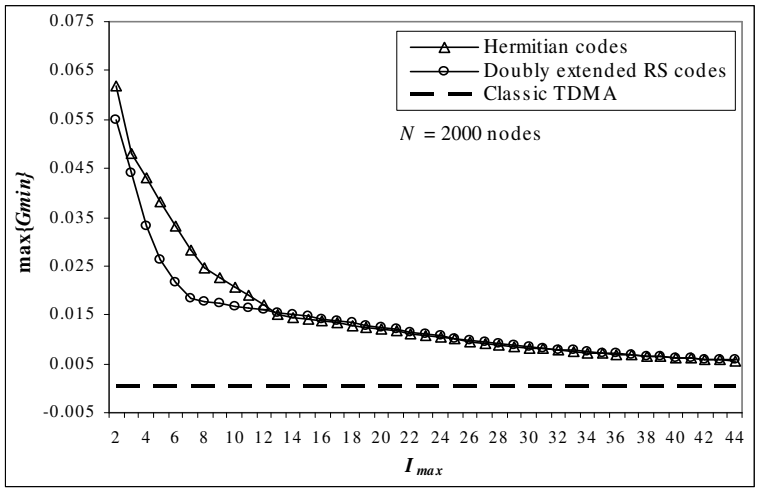

Figure 5.7 max{G_{min}} using DE-RS and Hermitian codes in a network of 2000 nodes

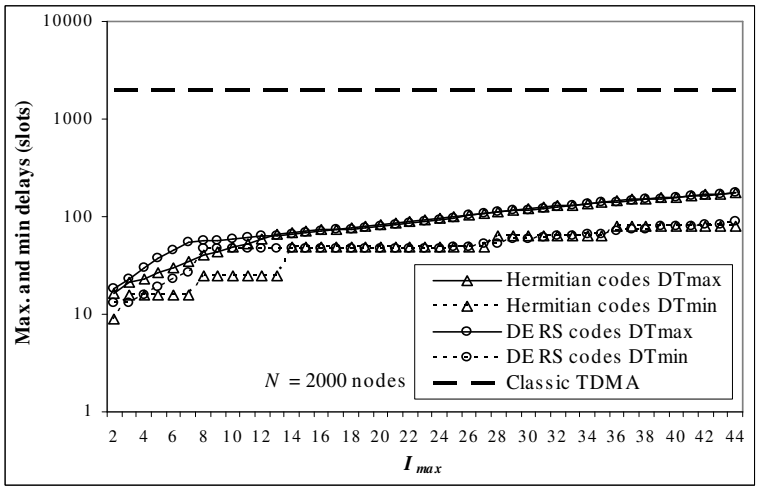

Figure 5.8 Maximum and minimum delays (in slots) using DE-RS and Hermitian codes in a network of 2000 nodes

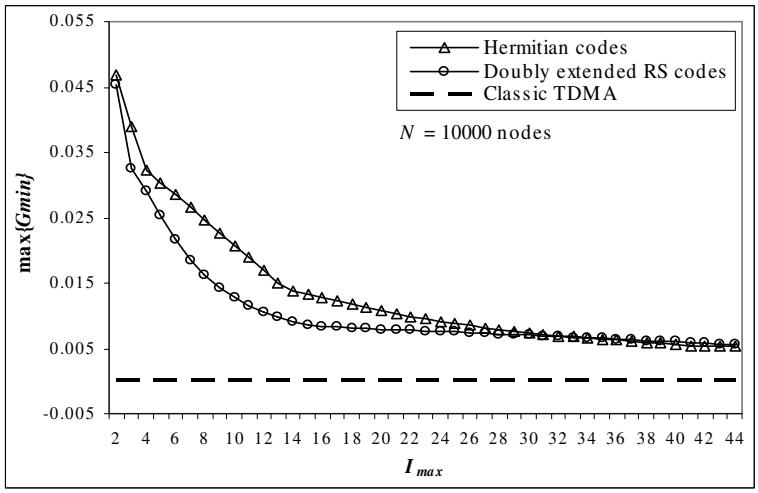

Figure 5.9 max{G_{min}} using DE-RS and Hermitian codes in a network of 10000 nodes

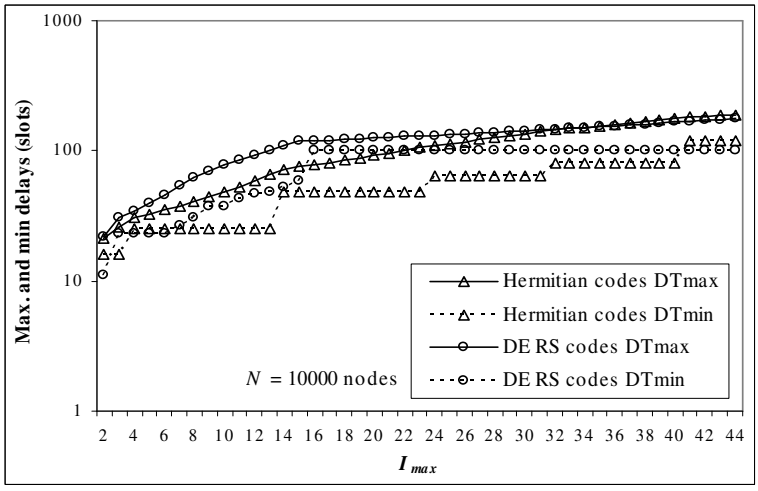

Figure 5.10 Maximum and minimum delays (in slots) using DE-RS and Hermitian codes in a network of 10000 nodes

thousands of wirelessly connected sensors are spread over large geographical areas. However, the maximum number of interferers of a given network must be clearly controlled if some performance advantage is desired. In any case, the control of $I_{\max}$ is advantageous for both constructions.

The reason why Hermitian codes possess a larger minimum throughput guarantee for certain $I_{\max}$ values can be explained as follows. Table 5.1 shows the parameters of some Hermitian and RS codes as a reference. Let us start by noting that the minimum Hamming distance for codes of rank one are the same as the RS codes of the same order, however, as the rank of the code increases beyond two, the Hermitian codes show smaller $n/d_{\min}$ ratios as observed in Table 5.1, and this is a crucial factor for the better performance of the Hermitian codes in certain regions. Take for instance, the RS and Hermitian codes of order q = 16 and rank k = 3. A doubly extended RS code of this characteristics will have n = 17, and $d_{\min} = 17-3+1 = 15$. The Hermitian code has $d_{\min} = 59$ with n = 64. Note that the Hermitian code is not MDS with these parameters since $d_{\min} < n-k+1$. However, the ratio $n/d_{\min}$ is smaller for the Hermitian code, and this makes $G_{\min}$ larger. The fact that $G_{\min}$ increases with codes with smaller $n/d_{\min}$ ratio can be seen more easily by looking at (5.3). Note that between two codes that have the same order (and for a given network with the same $I_{\max}$), the code that possesses the smaller $n/d_{\min}$ ratio will have larger $G_{\min}$ and hence it will be a preferred choice (i.e., regardless of it being an MDS code or not). The d_{min}/n ratio is known in error control coding terminology as the relative minimum distance of a code. When the number of nodes in the network is considerably increased, the rank of the codes will start to increase to satisfy the first inequality in (5.2) while maintaining a value of q considerably smaller than N (otherwise classic TDMA would have the same or better performance as shown before). This is the behavior observed in the codes that maximize $G_{\min}$ in Figures 5.3 through 5.10 when the number of nodes is increased.

Table 5.1 Parameters of some Hermitian and Doubly-extended RS codes

(The length *n* of the DE-RS code is q^2+1 to make comparisons fair)

q^2	*n* (Her)	*k*	dmin (Her)	n/dmin (Her)	dmin (RS)	n/dmin (RS)
4	8	1	8	1.00	5	1.00
4	8	2	6	1.33	4	1.25
4	8	3	5	1.60	3	1.67
4	8	4	4	2.00	2	2.50
9	27	1	27	1.00	10	1.00
9	27	2	24	1.13	9	1.11
9	27	3	23	1.17	8	1.25
9	27	4	21	1.29	7	1.43
16	64	1	64	1.00	17	1.00
16	64	2	60	1.07	16	1.06
16	64	3	59	1.08	15	1.13
16	64	4	56	1.14	14	1.21

Another important factor, discussed in [39], is how robust the scheduling approach is to errors in the estimation of the number of nodes in the network (N) and the maximum number of interferers ($I_{\max}$). Figures 5.11 and 5.12 show a sample of the variation in $G_{\min}$ as $I_{\max}$ varies around the designed value for both code constructions. The x-axis in Figures 5.11 and 5.12 represent the actual $I_{\max}$ denoted as $I_{\max}^{R}$ and the design parameter (that one that determines the code parameters to used based on the maximization of $G_{\min}$) is denoted as $I_{\max}^{D}$. The Hermitian code offers less sensitivity to a change in the number of nodes in the network in Figure 5.11. The reason is that the code used is the Her(125, 3, 25), which offers a total of 15,625 code-words, on the other hand, the RS code is the RS(33, 2, 32) when N^{D} =1000, which offers support for 1,024 users, and RS(60, 2, 59) when N^{D} =3000, which offers support for only 3,481 nodes with smaller relative minimum distance than the corresponding Her code. When $I_{\max}^{D}$ is increased both codes

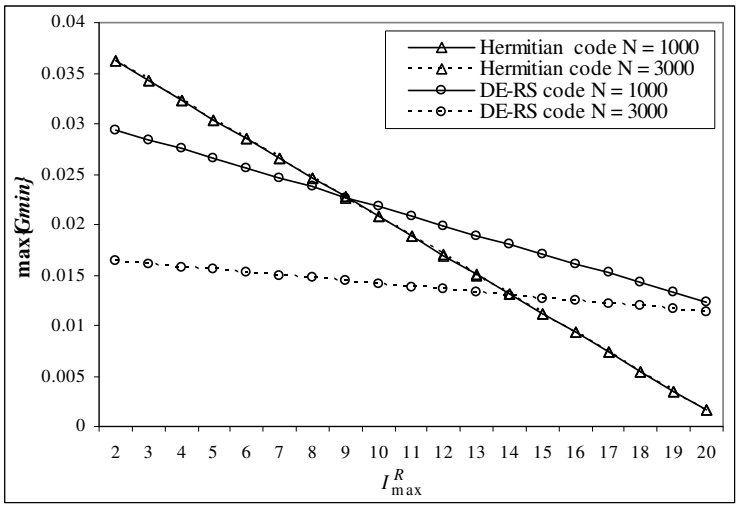

Figure 5.11 Variation in max{G_{min}} as $I_{\max}$ changes away from the design value $I^D_{\max} = 10$

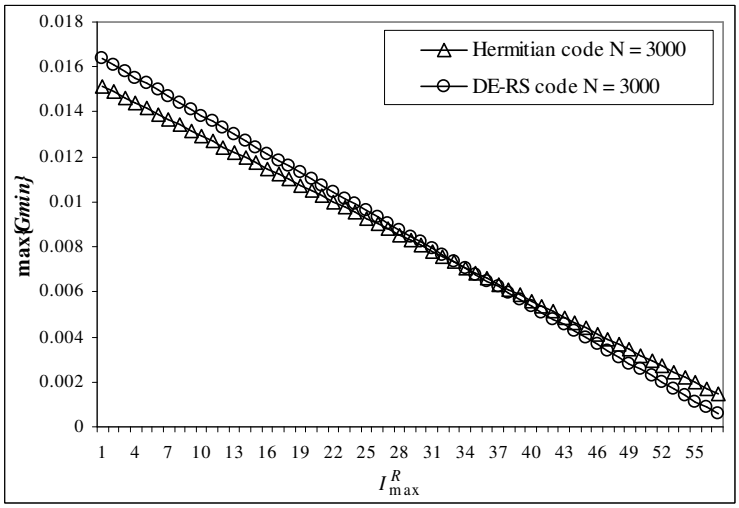

Figure 5.12 Variation in max{G_{min}} as $I_{\max}$ changes away from the design value $I^D_{\max} = 30$

show similar robustness as seen in Figure 5.12. The codes that maximize $G_{\min}$ in Figure 5.12 are Her(512, 2, 64) and RS(60, 2, 59). Based on these examples, we cannot generalize any findings about the relative robustness of both schemes to changes in the

design parameters; the particular code that maximizes $G_{\min}$ should be evaluated in terms of robustness for a particular design condition.

The advantage of the Hermitian codes (i.e., that of possessing a smaller $n/d_{\min}$ ratio than the RS codes when the rank is higher) could be utilized to increase the number of code-words available when the number of nodes increases without the need to increase the order of the code. Changing the order (q) of the code translates into a different frame size, which means a need to re-distribute all the code-word assignments per node in the entire network. If, however, a higher-rank code with an unchanged order is used, the original code-words assigned to the old nodes can still be used since they form a subset of the new code-set, and therefore, only additional code-word assignments will be needed for the newly arriving nodes. The latter is possible only in linear codes and represents a considerable practical advantage. Another potential advantage of Hermitian codes is that higher-rank codes with good performance could be used to generate an extremely large set of code-words (e.g., billions), which in turn means that a node would select randomly a code-word from this large set. The possibility of two nodes picking the same code-word would be almost zero assuming the number of code-words is much larger than the number of nodes. Therefore, this would represent a reduction in the overhead created by the assignment of unique code-words to nodes.

The next section explores the differences between a scheme that does not need the overhead of assigning unique slots or channels per node, and a code-based scheduling approach based on the Reed-Solomon code construction. Furthermore, average throughput performance of these scheduling schemes is investigated.

5.3 Comparative Evaluation of Code-based and Contention-based Scheduling Protocols

To the best of our knowledge we were the first to present an analytical comparison between a contention-based and a code-based scheduling protocol [52]. The comparison of two such schemes is important to understand more clearly the real advantages and limitations of a code-based approach. To start with, a code-based scheduling protocol has the important advantage of being able to guarantee a minimum throughput and packet delay performance. A random or contention-based scheduling approach is not able to claim such guarantees due to its random nature. However, in order to guarantee the minimum performance, a code-based approach requires, among other things, a unique assignment of code-words to nodes in the network, which can be a difficult problem in a dynamic and distributed wireless Ad Hoc network. In this section, we analytically compare the average performance of the slotted-ALOHA protocol with that of the OAs used in [40] and with a code-based approach based on RS codes. We describe a numerical procedure used to compare the performance of different code-based scheduling protocols in arbitrary topologies. We also define more clearly the different expected throughput metrics that can be used to evaluate a given code-based scheduling approach.

An expected throughput of a code-based approach based on OAs was derived in [45] for OAs of strengths two and three (i.e., the strength of an OA is analogous to the rank of a code). The expected throughput of a given node with i neighbors is defined in [45] as

$$\overline{G}_i^c = \sum_{w=0}^{n} (n-w) \frac{\binom{n}{w} C_i^w}{\binom{q^k - 1}{i}} \frac{1}{nq}. \tag{5.16}$$

Equation (5.16) is for a frame with n sub-frames and q slots per sub-frame, therefore we have a frame of nq slots; w represents the number of slots in which two code-words coincide in some specific positions out of the n possible positions (i.e., n is the length of the code-word). Given a code-word W, Equation (5.16) finds all the possible code-words that *collide* or coincide with W in w positions, where w takes values between 0 (i.e., no collisions at all, or Hamming distance of n) to n (i.e., all elements of the code-word collide, or Hamming distance of 0). The term (n-w) is the number of free or successful number of slot transmissions (i.e., slots without collision) for a particular value of w, which is multiplied by the probability of colliding in w positions, and finally divided by the frame size to obtain this expected throughput. C_i^w is the number of different ways in which i code-words coincide in w specific positions with the given code-word W (the union of all coincidences is taken). The result in (5.16) can also be obtained if the given code-word W is compared with all the possible groups of i code-words, then the union of the coinciding elements of all the i code-words with W is taken to compute the throughput. However, the latter procedure is, in certain cases, computationally intensive since the number of code-words in a given code can be very large. If, on the other hand, C_i^w can be computed, the value of $\overline{G}_i^c$ could also be conveniently obtained. The way to

compute C_i^w is given in [45], and can be obtained by using the following generating function

$$\phi(w;k,n,q)=\sum_{i=0}^{q^k-1} C_i^w x^i \ . \tag{5.17}$$

For the particular case in which $k = 2$ we have

$$\phi(w;2,n,q)=\left[(x+1)^{(q-1)}-1\right]^w\left[x+1\right]^{(q-1)(q+1-k)}. \tag{5.18}$$

The expected throughput as defined in (5.16) can be found by using Equation (5.17) or (5.18) in particular.

The slotted-ALOHA protocol is a contention-based scheduling protocol that requires slot synchronization. Slotted-ALOHA has the advantage of being relatively simple to implement when comparing it to a code-based approach, therefore it is interesting to compare their average performance.

Assuming that we have knowledge of the number of neighbors of a given node x[♣], we can state that the probability of successful transmission of x in slotted-ALOHA is as follows

$$\overline{G}_i^s = p(1-p)^i \ . \tag{5.19}$$

[♣] We use the term "number of neighbors" in this section to have some continuity with the language used in other related research work. However, a more accurate term would be number of interferers since nodes that are not direct one-hop neighbors could interfere as well in a wireless environment.

Where p is the transmission probability of a node, and i is the number of neighbors of the intended receiver. We can find the optimum value of the transmission probability p by differentiating (5.19), equating the result to zero and solving for p as follows

$$\frac{\partial \overline{G}_i^{\,s}}{\partial p} = (1-p)^i - i(1-p)^{i-1}\, p = 0 \Rightarrow p^o = \frac{1}{i+1}. \tag{5.20}$$

Substituting p^o in (5.19) we obtain

$$\overline{G}_i^{\,s} = \frac{1}{i+1}\left(1-\frac{1}{i+1}\right)^i. \tag{5.21}$$

Equation (5.21) is also the average-best-throughput of slotted-ALOHA assuming knowledge of i. Figure 5.13 shows $\overline{G}_i^{\,s}$ and $\overline{G}_i^{\,c}$ for a number of neighbors between 1 and 40. The $\overline{G}_i^{\,c}$ curves shown are for different sub-frame sizes (q values) between 3 and 27 and for strength two OAs with $n = q + 1$. The OA curves with smaller values of q decay more rapidly as the number of neighbors of the given node increases, however they have higher expected throughput with fewer number of neighbors (see Figure 5.13). As can be observed, the expected throughput of slotted-ALOHA is always larger than or equal to the expected throughput of OAs for all the number of neighbors considered. This result, however, does not necessarily mean that code-based scheduling approaches have *always* worse than or equal average performance than the classic slotted-ALOHA protocol.

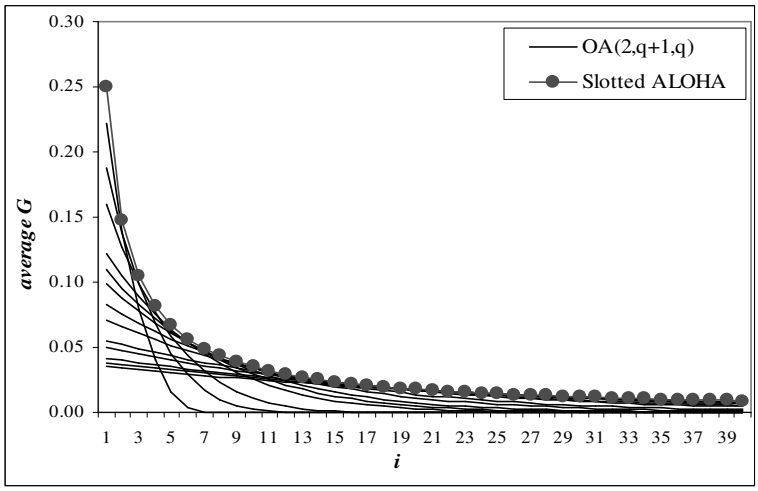

Figure 5.13 Expected throughput of OAs of strength two and slotted-ALOHA versus the number of neighbors of a node.

In order to fully compare both approaches, it is necessary to define expected throughput in several ways, and in fact, Equation (5.16) is one of the possible ways. In particular, we distinguish between sample and ensemble performance metrics. A sample performance metric is one that is obtained for a particular or *sample* network (i.e., a network with a specific number of nodes, maximum number of interferers, particular code-word assignments used etc), whereas an ensemble performance metric is a more general metric that is not directly concerned with the particulars of a given network, but on the other hand is focused exclusively on the mechanism under study.

A sample performance metric is a more meaningful metric in practical situations since what ultimately matters is the use of the proposed mechanism over a real network. However, an ensemble performance metric can be used to show, in a more general scenario and without ties to specific network conditions, the relative advantages or limitations between different mechanisms. Having introduced these concepts, we will now define some of these metrics by utilizing a simple example. We will describe a

numerical procedure that can be used to compute comparative performance evaluations in real networks.

Consider the network in Figure 5.14. A solid line denotes the possible communication links between two nodes. In the examples presented here the transmission range is equal to the interference range for simplicity. For instance, a transmission from node 1 to node 2 can only be interfered by transmissions from node 2 or node 3. However, the method presented here can be transparently applied when both ranges are different by considering the number of possible interferers, whether they are at transmission or at interfering range.

In Figure 5.14, $I_{\max} = 2$ and $N = 5$. If one wishes to use a Hermitian code for instance, it is found that the best code in terms of maximizing minimum throughput is Her(8,2,4), this is a Hermitian code of length 8, rank 2, and order 4. The code-words of Her(8,2,4) are shown in Table 5.2, note that $d_{\min} = 6$ (i.e., $d_{\min}$ can be easily computed as: *code length – max. number of zeros in any non-zero code-word*, this is due to the linearity of the code).

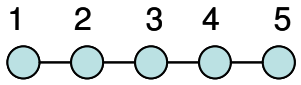

Figure 5.14 Wireless Ad Hoc Network with a line topology

Table 5.2 Code-words of Her(8,2,4)

CW1	0	0	0	0	0	0	0	0
CW2	0	0	1	1	2	2	3	3
CW3	0	0	2	2	3	3	1	1
CW4	0	0	3	3	1	1	2	2
CW5	1	1	1	1	1	1	1	1
CW6	1	1	0	0	3	3	2	2
CW7	1	1	3	3	2	2	0	0
CW8	1	1	2	2	0	0	3	3
CW9	2	2	2	2	2	2	2	2
CW10	2	2	3	3	0	0	1	1
CW11	2	2	0	0	1	1	3	3
CW12	2	2	1	1	3	3	0	0
CW13	3	3	3	3	3	3	3	3
CW14	3	3	2	2	1	1	0	0
CW15	3	3	1	1	0	0	2	2
CW16	3	3	0	0	2	2	1	1

For illustration purposes, assume we assign code-words (CWs) to nodes arbitrarily, but otherwise uniquely, as follows:

$$\begin{aligned} &\text{Node } 1 \rightarrow \text{CW5} \\ &\text{Node } 2 \rightarrow \text{CW2} \\ &\text{Node } 3 \rightarrow \text{CW12} \\ &\text{Node } 4 \rightarrow \text{CW8} \\ &\text{Node } 5 \rightarrow \text{CW7} \end{aligned} \tag{5.22}$$

We focus on each node and its possible interferers, and find its *success ratio*. Node 1's transmission gets affected by the transmissions of nodes 2 and 3 (assumed interferers). Therefore we investigate the interaction between node 1 and node 2 (1-2), and node 1 and node 3 (1-3).

Node 1: CW5 $\rightarrow$ 11111111

Node 2: CW2 $\rightarrow$ 00112233

Node 3: CW12 $\rightarrow$ 22113300

Number of (1-2) collisions: 2 in positions 3 and 4

Number of (1-3) collisions: 2 in positions 3 and 4

Since the two collisions in each pair happen at the same position with respect to CW1, the effective number of collisions is 2 (i.e., the union is taken). Therefore node 1 has 6 out of 8 attempts in which it is successful, and we define node's 1 *sample-average success ratio* using the code-word assignment combination in (5.22) (the j^{th} combination) as $SR_j^1 = 6/8$. We make the assumption that the nodes always have a packet to transmit.

Transmissions of node 2 get affected in one direction by node 1, and in the other with nodes 3 and 4. Therefore we investigate the sequences in the following two sets of pairs {(2-1)}, and {(2-3), (2-4)} separately

Right direction $\rightarrow$ Number of {(2-1)} collisions: 2 in positions 3 and 4

Left direction $\rightarrow$ Number of (2-3) collisions: 2 in positions 3 and 4

Number of (2-4) collisions: 2 in positions 7 and 8

Total of left direction: 4 collisions in positions 3, 4, 7, and 8.

We compute the success ratio in each direction and find the sample-average success ratio corresponding to the 2nd node and the j^{th} code-word combination assignment in (5.22) to be $SR_j^2 = \frac{1}{2}(6/8 + 4/8) = 0.625$. The latter average is assuming equally likely the

probability of node 2 (or any other node) to transmit in any direction, an imbalance in the direction probability can influence the value of the sample-average success ratio as defined above. However, we assume for simplicity an equally likely direction probability in order to have a common benchmark for the performance comparison of different codes, topologies, or scenarios. Furthermore, the maxima and minima of performance metrics can be computed regardless of the probability of transmitting in a particular direction. For instance, the sample maximum and minimum of the success ratio in the previous example is 6/8 and 0.5 respectively. Proceeding with our example, the sample-average success ratio and throughput for all nodes in the network are shown in Table 5.3.

Table 5.3 Sample-average success ratio and sample-average throughput for the nodes in Figure 5.14 using the code-word combination in (5.22)

Node	Sample-average success ratio	Sample-average throughput
1	0.750	0.18750
2	0.625	0.15625
3	0.750	0.18750
4	0.750	0.18750
5	0.500	0.12500

We define the *sample-average node throughput* (i^{th} node and j^{th} codeword combination) as $G_j^i = SR_j^i / q$. Note that $G_{\min} = 0.125$ for a Her(8,2,4), and $G_j^i \geq G_{\min} \ \forall i, j$.

The *sample-average network throughput* is defined as

$$G_N^j = \sum_{i=1}^{N} G_j^i , \tag{5.23}$$

where N is the number of nodes in the network, and the j index represents the use of a particular code-word assignment combination. For our particular example above we get $G_N^j = 0.844$.

Intuitively G_N^j is the number of successfully-utilized slots per time-slot unit in the entire network using the j^{th} code-word assignment combination. That is, if the information carried by a slot is called a *packet*, G_N^j is the number of packets successfully delivered in the entire network in an interval of time equivalent to a slot and using the j^{th} code-word assignment combination. G_N^j is equal to one in a classic TDMA scheme because a node is assigned a unique slot to transmit in the entire frame of size equal to the number of nodes in the network, therefore only one user is allowed to transmit in a given slot in the entire network.

In the following, N_c is the total number of codeword assignment combinations considered. The ensemble-average throughput of the network can also be computed as

$$\overline{G}_N = \frac{1}{N_c} \sum_{j=1}^{N_c} G_N^j \ . \tag{5.24}$$

The *ensemble-average node throughput* is

$$\overline{G}^i = \frac{1}{N_c} \sum_{j=1}^{N_c} G_j^i \ , \tag{5.25}$$

and the *ensemble-average success ratio per node* is

$$\overline{SR}^{i} = \frac{1}{N_c}\sum_{j=1}^{N_c} SR_j^i \ . \tag{5.26}$$

Note that $\overline{G}_N = \frac{1}{N_c}\sum_{j=1}^{N_c}\sum_{i=1}^{N} G_j^i = \sum_{i=1}^{N}\overline{G}^i$.

Other important metrics are the minimum and maximum ensemble-average network throughput, which are defined as $\min_j\{G_N^j\}$ and $\max_j\{G_N^j\}$ respectively. Note that the previous definitions are given in a numerical form. The numerical version of the definition in (5.16) is given in (5.25) and it is referred to as the *ensemble-average throughput per node*. The latter is a metric that encompasses all, or a sufficiently large number of code-word assignment combinations. The *sample-average throughput per node* is the true average throughput obtained in a particular network when a specific code-word assignment combination is used. The method presented above can be utilized to evaluate specific networks even under a mobility scenario.

The comparative result shown in Figure 5.13 tells us that slotted-ALOHA has better than or equal ensemble-average network throughput (EANT) than the OAs considered. OAs of higher strength have similar ensemble-average performance than strength two OAs [45]. The comparison with OAs bears a high degree of generality since the code-words of a great variety of codes can be used as the columns of OAs. Therefore, it is crucial to focus on improving the average performance of code-based approaches in order to compensate for the additional complexity of distributing unique codes, and ensuring an upper bound on the number of interfering nodes.

The ensemble-average metric used before hides information that could be used to compare both schemes in a more specific scenario. In particular, a real network does not always have all the possible code-words assigned to all the nodes in the network, instead, only a subset of all the code-words are in use. Take for instance a network of 100 nodes and $I_{\max} = 10$, in that case the best DE-RS code in terms of maximizing the minimum throughput is RS(12, 2, 11), this code has a total of 121 code-words, of which only 100 will be used. The situation gets more pronounced when $I_{\max}$ increases, for instance, for $I_{\max} = 20$ the best code is RS(38, 2, 37), which has a total of 1369 code-words.

We compare now the use of SE-RS codes with code-word selection against slotted-ALOHA. The performance metric used is the sample-average node throughput (SANoT). We select the code-words according to the following procedure,

Code-words assigned are identified as w_i, where $i = \{0,1,,\ldots,N-1\}$, and N is the number of nodes in the network. Code-words from the code-set (not assigned yet) are identified as W_j, where $j = 0,1,\ldots,q^k - 1$

1. Set $i = 0$ and start by picking a code-word W_j randomly for the given node (node 0) out of the q^k code-words. That is, $w_{i=0} = W_x$, where x is a uniformly distributed and discrete random variable that takes values from the set $J = \{0,1,,\ldots,q^k - 1\}$
2. Set $i = i + 1$
3. Pick a code-word W_j for the next node (node i), such that its Hamming distances d_{W_j,w_i} with respect to the already picked code-words satisfies:

$$\max_{W_j \neq w_i} \left\{ \frac{1}{i} \sum_{i=0}^{i-1} d_{W_j, w_i} \right\} \tag{5.27}$$

4. Set $i = i + 1$, and repeat 3 until $i > N - 1$

The code-word selection procedure above chooses new code-words that have maximum average Hamming distance to the code-words already chosen by the nodes in the network.

Table 5.4 shows the singly extended RS code parameters that maximize $G_{\min}$ constrained to the inequalities in (5.2) for different values of I_{max} and when 20 nodes are active in the network. Figure 5.15 shows a comparison between the SE-RS codes in Table 5.4 (i.e., $n = q$) using the code-selection algorithm against slotted-ALOHA (Equation (5.21)). The SANoT for a number i of neighbors is computed by selecting a code-word for the given node and comparing it with all the possible combinations of i-1 code-words (the code-words used by its neighbors).

The union of the collisions among the i code-words is used to compute the average success ratio of the given node, which in turn can be used to compute the SANoT. The initial code-word selected by the given node is taken to be all the code-words in the code-set in succession after which a final SANoT is computed. SE-RS codes are used to show that even when not using the best code-words (extended versions), the code-selection algorithm outperforms slotted-ALOHA.

Table 5.4 RS codes that maximize G_{min} with N =20 nodes

I_{max}	q	k
2	5	2
3	7	2
4	8	2
5	11	2
6	13	2
7	13	2
8	16	2
9	19	2
10	19	2
11	19	2
12	19	2
13	19	2
14	19	2
15	19	2

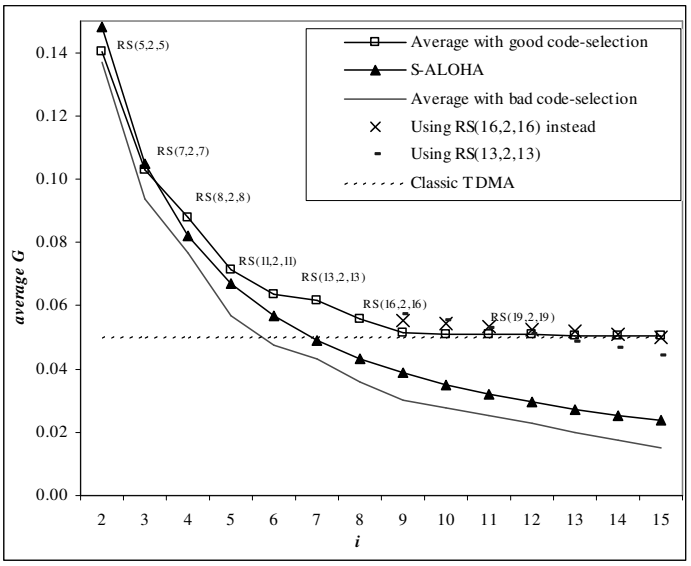

Figure 5.15 Average throughput performance using code-word selection and SE-RS codes

The minimum throughput of the codes used in Figure 5.15 is shown in Figure 5.16. Note that the minimum throughput shown is the *actual* value of minimum throughput.

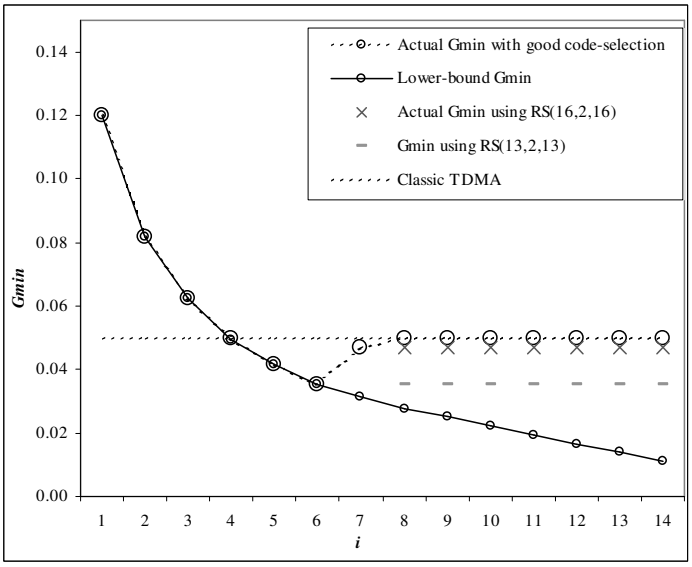

Figure 5.16 Gmin using code-word selection for codes in Figure 5.15

The actual value of minimum throughput is larger than or equal to the lower-bound $G_{\min}$ given by Equation (5.4). Both actual and lower-bound minimum throughput are shown in Figure 5.16. The performance improvement achieved by using code-word selection is more pronounced as the order of the code is increased to cope with larger neighborhood sizes. Figures 5.15 and 5.16 also show that codes that maximize (5.1) constrained to (5.2) are not necessarily the best in terms of average throughput. However, using other code parameters affects the minimum throughput guarantee by decreasing it considerably in some cases. The final choice of code parameters, however, will depend on the particular application and design requirements at hand.

An important practical issue that needs to be addressed in a code-based scheduling protocol is how to ensure that every node has a unique code-word. For a wireless Ad Hoc network this implies that every node should preferably pick its code-word(s) in a distributed manner, rather than using the services of a central entity responsible to assign the code-words in the entire network. The use of unique node identifiers, such as the MAC address or the IP address, has been suggested to accomplish this. Every node, based on its unique address, utilizes a Hash function to pick its code-word(s). If every node employs the same input sequence to the same encoder, the output sequence of code-words will be the same in every node, this is assuming every node has the same information necessary to specify the parameters of the encoder (i.e., $I_{\max}$ and N). The node could simply pick its code-word based solely on its unique address. If the code-selection algorithm is used, then the code-words generated by the encoder will pass through the additional selection step given by the code-selection algorithm. The code-selection algorithm can be made distributed if the initial code-word is, for instance, the

first code-word generated by the encoder given a common input sequence. Therefore, the code-selection algorithm will be as distributed as another method that does not select the code-words as long as all the nodes agree on using the same encoder, the same input sequence, and the same initial code-word to start the code-selection algorithm. Figure 5.17 depicts this idea.

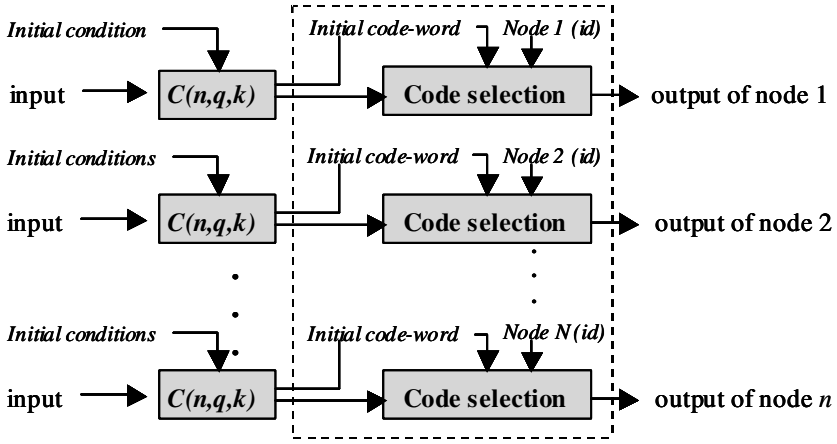

Figure 5.17 Code-selection algorithm for WAHNs

Every node has an encoder represented by $C(n, k, q)$, which is the same in every node of the network, and the same input sequence in $GF(q)$ with k digits. The output is a sequence of code-words that is equal in every node if all the encoders, initial conditions, and inputs are the same throughout the network.

A slight simplification of the code-selection algorithm can be achieved in order to reduce the processing power required to generate all the code-words of a code. The code-words in the first stage of Figure 5.17 (output of the encoder) can be generated in increasing order of their rank. That is, generate first the code-words corresponding to the

code with rank one (and the same order), continue with the code-words of rank two, and so on until the code-word of the given node is picked. The first set of code-words (the ones with rank one) need not be processed further through the code-selection algorithm. However, the code-words corresponding to the codes of rank two and higher need to be processed to ensure good mutual Hamming distance separation with code-words that have been already picked. Note that there is no need to implement the code-selection algorithm specified above if the set of code-words are generated in order of their rank. The mechanism above can eliminate the need for generating all the code-words in a code set.

5.4 Code-Contention-based Scheduling Algorithm

The previous sections highlighted the strengths and weaknesses of a code-based scheduling protocol. More remains to be done to further improve code-based scheduling. It is clear, however, that one of the key advantages of code-based scheduling is the fact that it has the potential to guarantee a minimum of performance regardless of the topological details of the network. There have been attempts to combine a code-based approach with contention-based approaches such as slotted-ALOHA in [52]. These hybrid approaches have resulted in schemes that take advantage of the fact that busy or idle slots remain in the same position of the frame if the network topology does not change for a considerable period of time. That is, a collision happens in the same slot as long as the topology of the network remains unchanged for a period of time and there are packets queued for transmission. This is due to the structured or deterministic way in

which the nodes access the medium. We would like a node to back-off when encountering collisions, or attempt to opportunistically transmit on an idle slot whenever the opportunity arises. Backing-off strategies may prove to be beneficial if done in a highly loaded system, while opportunistically transmitting in free-slots that are not dictated by the code-word may prove to be beneficial in a lightly loaded network (e.g., nodes with almost no traffic to transmit). Mechanisms based on receiver feedback and channel monitoring can be used to estimate the probability to back-off or attempt to transmit in a free-slot. The frame structure of Figure 5.1 can be expanded to include a time interval over which the receiving node will send an acknowledgement to the transmitting node. The transmitting node could then determine an update of the back-off probability based on the success or failure of its transmissions as it is frequently done in other distributed contention protocols. If a node monitors the channel and estimates that a particular slot is free, it could attempt to transmit in it. However, it is important to realize that incorporating contention-based schemes that opportunistically attempt to transmit on free-slots that are not dictated explicitly by the code-word, or even back-off only on the slots dictated by the code-word itself will increase the average performance of a node at the cost of reducing the capability of the scheduling approach to ensure a minimum performance guarantee. The minimum distance property among the code-words of a code is no longer valid once a node starts randomly trying in different slots. In other words, the determinism of the scheduling procedure will not be valid anymore. If a node backs-off in a specific slot dictated by its own code-word, the minimum performance can not be guaranteed either. A node can back-off in a slot after it encounters a collision and if the topology does not change the minimum performance guarantee will be kept. However, in

a mobile Ad Hoc wireless network the topology can change and the busy slot can turn into a free slot, and in that case the node that backed-off needs to quickly detect this situation if it wants to transmit and keep the desired minimum performance.

In this section we experiment with the latter approach (backing-off in slots dictated by the code-word) since we argue that it can effectively increase the average performance of a node with less impact on the minimum performance guarantees. First, however, we explain how a feedback mechanism can be incorporated into the original code-base scheduling procedure for the support of a back-off mechanism. Figure 5.18 shows a possible solution. It shows a time-guard between the different frame components in order to comply with the Tx-Rx turnaround times of the radios, which are considered half-duplex, an ACK per slot, and a beacon used for the network synchronization needed to construct the TDMA frame.

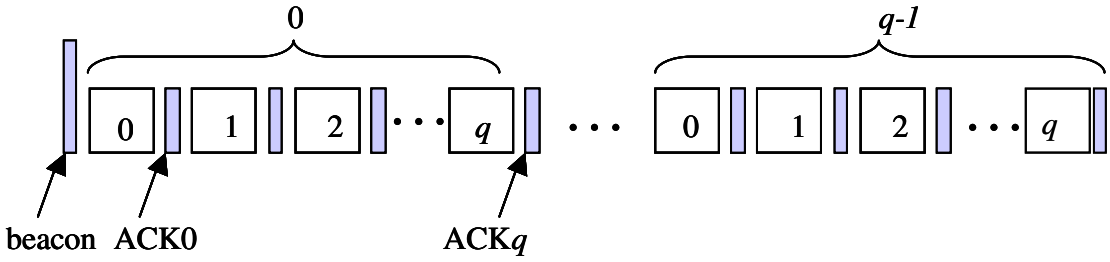

Figure 5.18 Frame structure with ACKs for back-off and beacons for network synchronization

An ACK coming from the intended receiver will be successful if the topology of the network does not change in the short period of time between the complete reception of a packet and the time it takes to send the corresponding ACK. Figure 5.19 depicts a network topology that can be used to explain the latter point.

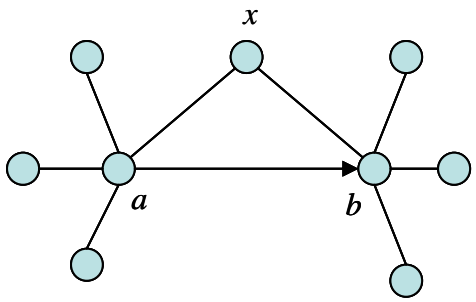

Figure 5.19 An ACK is successful if the packet is successful

Node *a* sends a packet to node *b*. Nodes *a* and *b* have their own neighbors, and also a common neighbor (node *x*). If node *a* is successful in the transmission of a packet to node *b* in a given slot, then it means that none of the neighbors of *b* transmitted a packet (including the common nodes) in that slot. Note that the neighbors of node *a* (excluding the common node *x*) might have transmitted a packet as well. If this packet was transmitted to node *a*, then it was unsuccessful since it collided with the transmission to node *b*, and no ACK will be sent from node *a*. This will ensure that the ACK coming from node *b* will not collide with an ACK transmission that could have come from node *a*. The other option is for a neighbor of node *a* to transmit to a node different to node *a* in that same slot. The only possibility for which a packet transmission would be successful (ensuring that the packet at node *b* is received successful as well) is that the neighbor node transmits to a node that is not a neighbor of either nodes *a* or *b*. If this occurs then the ACK from node *b* to node *a* will not collide with the ACK from that second-hop node to the neighbor of node *a*. Therefore, as long as a packet is received successfully, the ACK will also be received successfully by the sending node.

We proceed now with a more detailed description of the code-contention-based scheduling algorithm.

The time-frame is shown in Figure 5.18. We define two ways to schedule the transmissions, 1) a pure code-based and 2) a hybrid code-contention-based scheduling approach.

Pure code-based scheduling: In the pure code-based approach each node transmits in the slot indicated by the code-word assigned to it regardless of the traffic being forwarded. The i^{th} node uses its code-word's x^{th} digit (i.e., $C_i(x)$) to select the time-slot in sub-frame x (the counting of digits and sub-frames is assumed to start from zero). This is the original method without any modification.

Code-contention-based scheduling: The Code-contention-based scheduling approach is referred to as hybrid scheduling in what follows. Hybrid scheduling is simple in principle. A transmitting node follows its code-word to transmit its packets and expects an ACK for each transmitted packet. In the absence of the ACK from the intended receiver the transmitting node will back-off with probability $p = 1/I_{\max}$ in that slot. That is, the access mechanism becomes slotted-ALOHA in those slots in which a node detects collisions, and the original code-based scheduling approach in those slots in which all transmissions have been successful. The hybrid approach needs to be able to detect collisions and also detect situations in which collisions are resolved because interfering nodes moved, stopped transmitting packets, or left the network. Therefore the above probability p is actually adaptive in a more advanced variation of the hybrid approach. However, for the analysis that follows we will assume that the number of interferers in a neighborhood is fixed in the period of observation and that nodes always have a packet to

transmit. This allows us to take the fixed value of $p = 1/I_{\max}$ for simplicity and show the potential improvement obtained by using the hybrid approach.

Figure 5.15 shows the average throughput obtained with pure code-based scheduling protocol when the code-selection algorithm is used and there are twenty nodes in the network. Figure 5.20 shows the average number of slots in which collisions happen versus the number of interfering nodes for Figure 5.15. In order to compute the average number of unsuccessful slots, the following equation is used,

$$U = q - G_{cs} \cdot q^2, \tag{5.28}$$

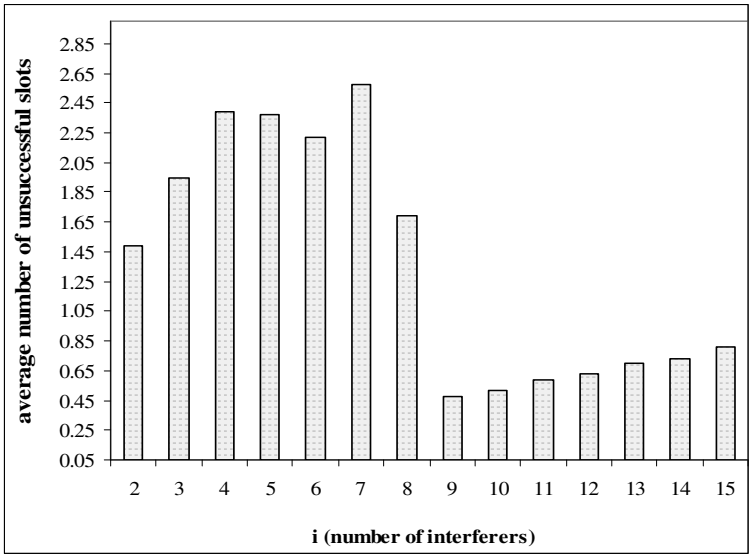

Figure 5.20 Average number of unsuccessful slots versus the number of interferes of Figure 5.15

where U denotes the average number of unsuccessful slots, q is the order of the code and G_{cs} is the throughput of code-based scheduling using code-selection. Figure 5.20 is

obtained after the values plotted in Figure 5.15 are used. Having the data on Figure 5.20 it is possible to find the average throughput of the hybrid approach as follows,

$$G_{hybrid} = \frac{U \cdot G_{s-ALOHA}}{q^2} + G_{cs}\,, \tag{5.29}$$

where G_{hybrid} is the throughput of the hybrid approach and $G_{S-ALOHA}$ is the throughput of slotted-ALOHA given in Equation (5.21). Figure 5.21 plots G_{hybrid} and G_{cs} versus i (number of interferers). As can be observed, the improvement is more pronounced when the number of interferers is low. Even though the improvement does not seem substantial, looking at Figure 5.20 one can infer that the code-based scheduling algorithm is already rather efficient since it has a small number of unsuccessful slots for each of the cases considered.

5.5 Summary

Code-based scheduling is presented as a generalization to the topology-transparent scheduling protocol proposed by Chlamtac and Farago [17]. It is pointed out that *any* code can be used for scheduling purposes, and that focus should be directed towards existing or new code constructions that can provide better average and minimum performance guarantees. Two code constructions were evaluated. In particular it is found

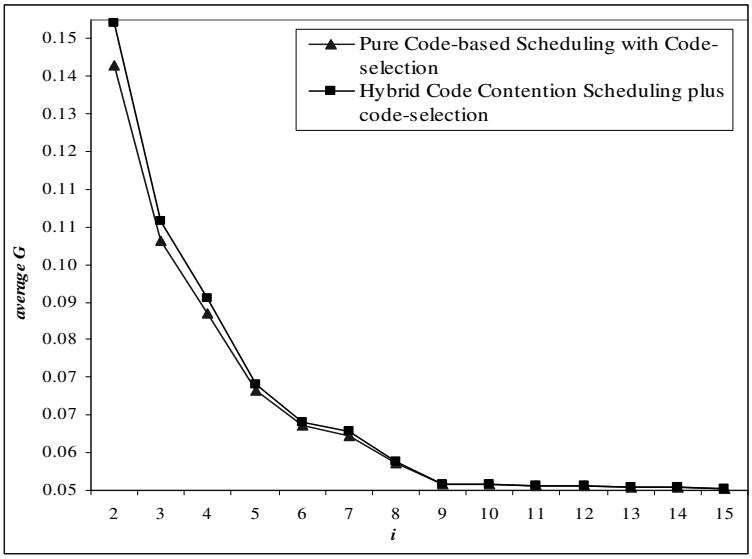

Figure 5.21 Average throughput of the hybrid and pure code-based scheduling approaches

that a code with smaller relative minimum distance will perform better in terms of guaranteeing more throughput and less delay when its code-words are used as a scheduling sequence. Hermitian codes are found more suitable for large and sparse networks of thousands of nodes. In this work we focus on tangible code constructions to find concrete performance evaluations, and we utilize the terminology of coding theory to define performance metrics and a framework for analysis.

The need for improved average performance of code-based scheduling is highlighted and a method for code-word selection is proposed. Our results show good potential for the improvement of the performance of code-based scheduling protocols through the use of code-word selection. It is important to note that in our previous average-throughput results we assumed immediate acknowledgement of successful packet reception by the intended receiver. The work in [52] describes and discusses an implementation of an acknowledgement procedure suitable for code-selection scheduling protocols in the

context of Time Spread Multiple Access [17]. Code-based scheduling protocols have the potential to out-perform contention based scheduling protocols in average and minimum guarantee performance through a suitable code-selection procedure.

It is important to note that code-based scheduling does not have a good performance in a single-hop topology as shown in equation (5.15). Furthermore, better alternatives can be found if the topology of the network does not change (i.e., the nodes are fixed). Therefore, code-based scheduling is more suitable for wireless Ad Hoc networks with *multi-hop* topology and mobility.

A code-contention-based scheduling algorithm was proposed that further improves the average throughput performance of code-based scheduling with code-selection. A node implementing code-contention-based scheduling backs-off probabilistically in those slots in which collisions are encountered. Even though the performance enhancement obtained with code-contention-based scheduling seems marginal we argue that a greater enhancement should be observed in real-traffic conditions. Additionally, the minimum performance guarantee of code-based scheduling is better maintained by limiting a node to transmit in those slots dictated by its code-word(s). Opportunistic techniques that try to transmit in any slot of the frame, such as the one proposed in [52] destroy this major advantage of code-based scheduling.

Chapter 6

Conclusion and Future Work

6.1 Conclusions

We have proposed a novel network synchronization algorithm based on the concept of mutual network synchronization and clock-sampling that has the potential of achieving accuracies in the order of few micro-seconds for multi-hop or single-hop wireless networks with minimum or no infrastructure. The latter is achieved with equal or less overhead than the IEEE 802.11 Timing Synchronization Function (TSF). Table 6.1 summarizes the major differences between Clock-Sampling Mutual Network Synchronization (CSMNS) and the TSF.

Different procedures can be added to CSMNS in order to reduce its overhead thanks to its non-hierarchical design. We have proposed two ways to accomplish this, in particular, the rotating master node CSMNS, and the randomization of the beacon transmissions. Furthermore, we have identified the fact that the non-hierarchical property of CSMNS eliminates the beacon transmission uncertainties that plague other network synchronization approaches. A very accurate beacon transmission time can be known in advance due to the fact that every node has the same influence over the synchronization of the network. Additionally, we have analyzed a procedure that could be employed to improve the TSF and that we refer to as the modified TSF.

Table 6.1 Comparison of CSMNS and TSF

	TSF	CSMNS
Accuracy (order of)	**≈10e-3**	**≈10e-6**
Scalability	**Max. 10-20 nodes**	**Hundreds**
Stability	**Yes**	**By design**
Complexity	**Low**	**Low**
Overhead w.r.t the TSF	**-**	**Less than or equal**
Multi-hop capability	**Yes**	**Yes**
Convergence	**No**	**Yes**

In the modified TSF every node transmits its own beacon even after successfully receiving a beacon from another node or if its own timestamp is later than the one received. The latter procedure can be used when the overhead caused by the extra beacon transmissions are worth the approximately double accuracy achieved compared to TSF. However, the procedure inherits the hierarchical and marginally convergent characteristics of the TSF that are eliminated when using CSMNS. Both CSMNS and the TSF were evaluated in multi-hop and single-hop scenarios.

We have proposed a new way to look at the MAC scheduling problem based on the idea of using codes traditionally employed to correct and/or detect errors in the information transmitted over a noisy channel. The realization that any of the large number of code constructions available can be used for scheduling in a WAHN represents a generalization to the topology-transparent scheduling protocols proposed by Chlamtac and Farago for multi-hop packet radio networks [17], and more recently by

Syrotiuk et al., using orthogonal arrays (OAs) [40]. In particular, it is realized that the coding theorists goal of finding codes with the largest possible distance among its constituent code-words in an ever smaller dimensional space is analogous to the goal of finding separate yet efficient ways for the nodes of a network to transmit in a multiple access system.

Chlamtac's protocol construction is identified as being the same as the one used to construct a singly-extended Reed-Solomon code. This protocol restricts the frame size to be the square of the Galois field dimension used to define the unique polynomials in every node. However, the coding theory point of view allows us to realize that it is possible to have different frame sizes and hence more flexibility in the design of such protocols. In particular, Reed-Solomon codes can always be doubly-extended, shortened, and, in some cases, triply-extended, therefore the frame size is not restricted to be the one specified above due to the possible variations a given code can have. Furthermore, the use of different code variations influences the minimum performance guarantee of the scheduling approach as shown in Chapter 5.

Code-based scheduling represents a more global generalization to topology-transparent scheduling protocols than the one proposed in [40] and, perhaps more importantly, it provides a point of view that highlights the importance of focusing on the performance of different code constructions separately. The authors in [40] proposed a generalization based on OAs. However, the use of OAs intrinsically diminishes the importance of the specific constructions used to create them since many code constructions can produce the columns of an OA. Which code is better, and under what conditions, is an important question that needs to be answered. Additionally, simple-to-

construct codes are preferred when implementing these scheduling approaches, particularly for distributed systems.

The fact that any code (e.g., linear, non-linear, variations of the same code, or different codes altogether) can be employed for scheduling purposes opens the question of what makes a code better than others. In terms of minimum performance guarantee, we have found that codes with larger relative minimum distance can guarantee more throughput and less delay performance. The latter explains, for instance, why a doubly-extended Reed-Solomon code has better minimum performance guarantees than a singly-extended one. In line with the previous result, we gave analytical results on a procedure to choose the parameters of a code that improves over the optimal presented in [39].

We conclude that our quest for good codes must be directed toward code constructions with good relative minimum distance properties if better guarantees are desired. A family of codes with good relative minimum distance is represented by the Algebraic Geometric (AG) codes. We have compared Reed-Solomon codes to codes based on Hermitian curves (Hermitian codes) in terms of minimum performance guarantee. We have found that Hermitian codes are more suitable than Reed-Solomon codes for large and sparse networks. Networks that fall in the latter framework could be, for instance, future micro-sensor networks. Contrary to intuition, the maximum distance separable (MDS) property of a code does not in itself guarantee a better minimum performance. Hermitian codes are not MDS. However, they have better minimum performance than the MDS Reed-Solomon codes in some cases due to their larger relative minimum distance.

Average performance metrics are also of crucial importance in a communications network. We analytically compared the average throughput performance of code-based scheduling to that of slotted-ALOHA. The latter comparison is important since it is expected that the complexity of code-based scheduling over a contention-based scheduling approach, such as slotted-ALOHA, will be balanced by a better overall performance. Even though code-based scheduling has the unique potential advantage of being able to guarantee a minimum performance, it is expected that its average performance will also be higher than the one achieved by a contention-based scheduling approach. However, this is not the case if the code-words are not chosen appropriately. In particular, the average throughput performance of a code-based scheduling protocol can fall below the one of slotted-ALOHA if the code-words are not far enough from one another in Hamming distance. We have proposed a code-word selection algorithm that chooses code-words with good mutual distance separation. Our code-word selection algorithm improves the average throughout performance of a code-based scheduling protocol over that of slotted-ALOHA when the number of nodes in the network is less than the number of code-words available.

An important issue of code-based scheduling is the fact that every node must have a unique code-word(s) assigned in order to guarantee a minimum performance. One way to solve this problem in a distributed way is for every node to generate the code-words independently based on a common encoder, and a common input sequence to the encoder. That is, as long as the encoder and its input is the same in every node of the network, the output sequence of the encoder (code-words) will also be the same in every node. A node will then pick its code-word based on its unique identifier (such as a MAC

or IP address). This has the potential problem that the process to generate all the code-words would require substantial processing power if the number of code-words is large due to a code with a large dimension. However, in most cases the dimension of the code will not be in the order of thousands or even hundreds since a network of that dimension or density is not yet a reality. The same procedure can be implemented when the code-selection algorithm is used. Therefore, there is no extra burden added when implementing the code-selection algorithm over a random code-word selection as far as the code-assignment problem is concern. Note that a node can stop generating code-words as soon as it finds its own based on its unique identifier.

6.2 Future Work

As for future work in the area of network synchronization, we are studying the possibility to adapt the frequency of beacon transmissions according to the dynamics of the network in CSMNS. That is, opportunistically decrease the frequency of beacon transmissions when the node mobility or the wireless medium impairments decrease, and increase it otherwise. This would represent an improvement in the use of the space and processing power resources. Additionally, dynamic adjustment of the permission probabilities in order to avoid the need for estimating the number of nodes in the network will be pursued. Other areas of interest in the area of network synchronization include the performance evaluation of the TSF and CSMNS in a mobile environment with a more realistic wireless channel. It is expected that both mechanisms will degrade their performance due to the higher beacon error rate. However, this problem has not been

quantified before and it would be interesting to evaluate which method, if any, is affected more by these impairments. Another interesting problem is that of quantifying the time needed to acquire initial synchronization due to the fact that in some applications the nodes may leave and join the network quite rapidly.

We have studied in this thesis a particular way of linking coding theory with the MAC layer of a communication network. However, more needs to be done in this area. In particular, due to the fact that decoding is not needed in code-based scheduling it is important to address the problem of finding new unexplored code constructions that can provide better access-to-the-medium performance for the nodes of a network, such as non-linear code constructions or other linear codes. Codes that are deemed complicated in error control coding due to their decoding complexity can, however, be used in code-based scheduling. Additionally, entirely new code constructions tailored to the code-based scheduling approach will be explored. Future work could also address how to make code-based scheduling less dependent on network synchronization, improve further its average performance, make it less dependant on the information it needs, and more resilient to uncertainties. While we have evaluated the performance of well-known code constructions, we believe it is necessary to pursue more general work in this area. In particular, it is important to find an optimum frame size and the optimum code parameters and characteristics that can offer the best possible average throughput. A possible way of achieving this would be to re-define the format of the frame, and also to look for longer codes with large minimum distances (e.g., product codes).

We are also pursuing a complete network simulation of code-contention-based scheduling with more realistic traffic models, mobility, and wireless channel. Code-

contention-based scheduling will be compared with the CSMA/CA protocol in a simulation in which both MAC protocols will be part of a complete protocol stack (i.e., physical, network, transport, and application layers). Other ideas for future work include the use of directive antennae combined with code-based scheduling to reduce the numbers of interferes and improve throughput performance, the study of non-linear codes for code-based scheduling, the use of multiple codes in a single node in order to provide DiffServ type of service support, and finally the incorporation of low-overhead topology information into the code-based scheduling approach. A possible way to achieve this would be for a given node to passively observe how busy the medium is in its surroundings and adjust the way in which it access the medium, or select the best possible code-word based on a estimation of the code-words used by its neighbors.

References

[1] C-Y. Chong and S. P. Kumar, "Sensor networks: evolution, opportunities, and challenges," *Proceedings of the IEEE*, vol. 91, No. 8, pp. 1247-1256, August 2003.

[2] IEEE Std. 802.11-1999, "Wireless LAN medium access control (MAC) and physical layer specification,".

[3] IEEE Std. 802.16-2004, "Air interface for fixed broadband wireless access systems".

[4] IEEE Std. 802.15.1-2002, "Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs)".

[5] IEEE Std. 802.15.4-2003, "Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs)".

[6] D. J. Goodman, R. A. Valenzuela, K. T. Gayliard, and B. Ramamurthi, "Packet Reservation Multiple Access for Local Wireless Communication," *IEEE Transactions on Communication*, vol. COM-37, pp. 885-890, August 1989.

[7] Y. Du, Y. Bao, J. J. Garcia-Luna-Aceves, " A History-based Scheduling Protocol for Ad Hoc Networks," *in Proceedings of the 12th International Conference on Computer Communications and Networks*, 2003.

[8] A. Tzamaloukas and J.J. Garcia-Luna-Aceves, " A Channel-hopping Protocol for Ad Hoc Networks," *in Proceedings of the 9th International Conference on Computer Communications and Networks*, 2000.

[9] Z. Chen and A. Khokhar, "Self Organization and Energy Efficient TDMA MAC Protocol for Wireless Sensor Networks," *in Proceedings of the IEEE Conference on Sensor and Ad Hoc Communications and Networks*, 2004.

[10] H. Singh and S. Singh, "A MAC Protocol based on Adaptive Beacon Forming for Ad Hoc Networks," *in Proceedings of the 14th IEEE Personal, Indoor and Mobile Radio Communication Conference*, 2003

[11] Y-C. Hu, A. Perrig, and D. Johnson, "Packet leashes: A defense against wormhole attacks in wireless ad hoc networks," in *Proceedings of the IEEE Infocom*, 2003, pp. 1976-1986.

[12] Douglas M. Considine, Ed., *Van Nostrand's scientific encyclopedia*. New York: McGraw-Hill, ISBN 0-442-01864-9, 1995.

[13] P. Kartaschoff, "Synchronization in digital communications networks," *Proceedings of the IEEE*, vol. 79, No. 7, pp. 1019-1028, July 1991.

[14] S. Knappe, P. Schwindt, V. Shah, L. Hollberg, J. Kitching, L. Liew, and J. Moreland, "Microfabricated atomic frequency references," in *Proceedings of the IEEE International Frequency Control Symposium*, 2004, pp. 87-91.

[15] W.C Lindsey, F. Ghazvinian, W. Hagmann, and K. Dessouky, "Network synchronization," *Proceedings of the IEEE*, vol. 73, No. 10, pp. 1445-1468, October 1985.

[16] D. Bertsekas and R. Gallager, *Data Networks*. Upper Saddle River, NJ: Prentice Hall, ISBN 0-13-200916-1, 1992.

[17] I. Chlamtac and A. Farago, "Making transmission schedules immune to topology changes in multi-hop packet radio networks," *IEEE/ACM Transactions on Networking*, vol. 2, No.1, pp. 23-29, February 1994.

[18] S. Bregni, "Clock stability characterization and measurement in telecommunications," *IEEE Transactions on Instrumentation and Measurement*, vol. 46, No. 6, pp. 1284-1294, December 1997.

[19] D.W. Allan, "Clock characterization tutorial," in *Proceedings of the 15th Annual Precise Time and Time Interval (PTTI) Applications and Planning Meeting*, 1983.

[20] Hu and S. D. Servetto, "Asymptotically optimal time synchronization in dense sensor networks," *in Proceedings of the 2nd ACM Workshop on Sensor Networks and Applications* (WSNA), 2003, pp. 1-10

[21] J. Nelson, L. Girod, and D. Estrin, "Fine-Grained network time synchronization using reference broadcast," in *Proceedings of the 5th Symposium on Operating Systems Design and Implementation* (OSDI'02), 2002, pp. 147-163.

[22] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Communications of the ACM*, vol. 21, No. 7, pp. 558-565, July 1978.

[23] K. Römer, "Time synchronization in Ad Hoc networks," in *Proceedings of the ACM MobiHoc conference*, 2001, pp. 173-182.

[24] Gersho and B. J. Karafin, "Mutual synchronization of geographically separated oscillators," *Bell Systems Technical Journal*, vol. 45, pp.1689-1904, December 1966.

[25] E. Sourour and M. Nakagawa, "Mutual decentralized synchronization of intervehicle communications," *IEEE Transactions on Vehicular Technology*, vol. 48, No. 6, pp. 2015-2027, November 1999.

[26] B. Sundararaman, U. Buy, and A. D. Kshemkalyani, "Clock synchronization for wireless sensor networks: A survey," *Ad-Hoc Networks*, 3(3), pp. 281-323, May 2005.

[27] K. Arvind, "Probabilistic Clock Synchronization in Distributed Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, No. 5, pp. 474-487, May 1994.

[28] F. Cristian, "Probabilistic Clock Synchronization," *Distributed Computing*, vol. 3, pp. 146-158, Springer-Verlag, 1989.

[29] H. Dai and R. Han, "TSync: A Lightweight Bidirectional Time Synchronization Service for Wireless Sensor Networks," *ACM SIGMOBILE, Mobile Computing and Communications Review*, vol. 8, No. 1, pp. 125-139, January 2004.

[30] S. Ganeriwal, R. Kumar, and M. Srivastava, " Timing-Sync Protocol for Sensor Networks," *in Proceedings of the first International Conference on Embedded Networked Sensor Systems*, 2003

[31] A.D. Kshemkalgani, "The Power of Logical Clock Abstractions," *Distributed Computing*, vol. 17, No. 2, pp. 131-150, Springer-Verlag, 2004.

[32] M. D. Lemmon, J. Ganguly, and L. Xia, " Model-based Clock Synchronization in Networks with Drifting Clocks," *in Proceedings of The Pacific RIM International Symposium on Dependable Computing*, 2000.

[33] Q. Li and D. Rus, "Global Clock Synchronization in Sensor Networks," *in Proceedings of The IEEE Conference on Computer and Communications* (INFOCOM), vol. 1, pp. 564-574, 2004.

[34] D. L. Mills, "Internet Time Synchronization: The Network Time Protocol," *IEEE Transactions on Communication*, vol. 39, No. 10, pp. 1482-1493, October 1991.

[35] M. Mock, R. Frings, E. Nett, and S. Trikalioitis, "Continuous Clock Synchronization in Wireless Real-time Applications," *in Proceedings of the 19th IEEE Symposium on Reliable Distributed Systems*, pp. 125-133, 2000.

[36] A. Olson and K. Shim, "Fault-tolerant Clock Synchronization in Large Multicomputer Systems," *IEEE Transactions on Parallel and Distributed Computing*, vol. 5, No. 9, pp. 912-923, September 1994.

[37] L. Huang and T-H Lai, "On the scalability of IEEE 802.11 Ad Hoc networks," in *Proceedings of the ACM MobiHoc conference*, 2002, pp. 173-182.

[38] Carlos H. Rentel and Thomas Kunz, "A clock-sampling mutual network synchronization algorithm for wireless ad hoc networks," in *Proceedings of the IEEE Wireless Communications and Networking Conference*, 2005.

[39] Ji-Her Ju and Victor O. K. Li, "An optimal topology-transparent scheduling method in multi-hop packet radio networks," *IEEE/ACM Transactions on Networking*, vol. 6, No. 3, pp. 298-306, June 1998.

[40] V.R. Syrotiuk, C. J. Colbourn, and A. C. H. Ling, "Topology-transparent scheduling for MANETs using orthogonal arrays," in *Proceedings of the International Conference on Mobile Computing and Networking*, 2003, pp. 43-49.

[41] G. Eniarsson, "Address assignment for a time-frequency-coded, spread-spectrum system," *The Bell Systems Technical Journal*, vol. 59, No. 7, pp. 1241-1255, September 1980.

[42] G. Solomon, "Optimal frequency hopping sequences for multiple-access," in *Proceedings of the Symposium of Spread Spectrum Communications*, vol. 1, AD-915 852, 1973, pp. 33-35.

[43] R.J. McEliece, *Finite fields for computer scientist and engineers*. Kluwer Academic, ISBN 0-898-38191-6, 1987.

[44] C. J. Colbourn, V. R. Syrotiuk, and A. C. H. Ling, "Steiner systems for topology-transparent access control in MANETs," in *Proceedings of the International Conference on Wireless and Ad Hoc Networks*, 2003, pp. 247-258.

[45] C. J. Colbourn, A. C. H. Ling, and V. R. Syrotiuk, "Cover-free families and topology-transparent scheduling for MANETs," *Designs codes and cryptography*, vol. 32, No. 1-3, pp. 65-95, May 2004.

[46] A.S. Hedayat, Neil J. A. Sloane, John Stufken, *Orthogonal Arrays : Theory and Applications*. Springer-Berlag, ISBN 0-387-98766-5, 1999.

[47] F.J. MacWilliams and N. J. A. Sloane, *The theory of error-correcting codes*, North Holland, ISBN 0-444-85193-3, 1977.

[48] O. Pretzel, *Codes and algebraic curves*, Oxford science publications, ISBN 0-198-50039-4, 1998.

[49] E. R. Berlekamp, "Goppa Codes," *IEEE Transactions on Information Theory*, vol. IT-19, pp. 590–92, Sept. 1973.

[50] Carlos H. Rentel and Thomas Kunz, "On the average-throughput performance of code-based scheduling protocols for wireless ad hoc networks," *presented at the ACM MobiHoc Conference*, Urbana-Champaign, USA, 2005.

[51] R. Krishnan and J. P. G. Sterbenz, "An evaluation of the TSMA protocol as a control channel mechanism in MMWN," *BBN Technical Memorandum* No. 1279. Job No. 11982000, April 2000.

[52] K. Oikonomou and I. Stavrakakis, "Analysis of a probabilistic topology-unaware TDMA MAC policy for Ad Hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, No. 7, pp. 1286-1300, September 2004.