IEEE 1588 STYLE SYNCHRONIZATION OVER A WIRELESS LINK

A Thesis

Presented to

The Graduate Faculty of The University of Akron

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

Hamza Abubakari

December, 2008

IEEE 1588 STYLE SYNCHRONIZATION OVER A WIRELESS LINK

Hamza Abubakari

Thesis

Approved:

Advisor
Dr. Shivakumar Sastry

Committee Member
Dr. Nathan Ida

Committee Member
Dr. James E. Grover

Accepted:

Department Chair
Dr. Alex De Abreu Garcia

Dean of the College
Dr. George K. Haritos

Dean of the Graduate School
Dr. George R. Newkome

Date

ABSTRACT

Networked embedded systems, which rely on inexpensive nodes (hardware) and interact in a peer-to-peer manner over wireless links, offer new opportunities for systems architecture and design in a variety of domains. The local clocks in such nodes present relatively large clock offsets at the application level. Achieving time synchronization across such nodes, however, remains a challenge. The IEEE 1588 time synchronization protocol specifies how such synchronization can be achieved over wired networks. The wireless domain further exacerbates the problem of achieving time synchronization because of high packet losses and low bandwidth. *This thesis presents the design and implementation of a technique for synchronizing clocks, over a wireless link, of a pair of resource-constrained nodes.* This is a software-only implementation, in the sense that there is no special hardware required to support the technique. This design builds on a prior technique that compensated clock offset using a conventional digital filter and a Proportional-Integral (PI) controller using IEEE 1588 messages over a wired network. To mitigate the effects of packet-losses in the wireless environment this design tracks and compensates for skew, which is the rate of change of the offset. Because the skew was determined to be gaussian distributed, a linear Kalman filter was used to track the skew. This filtered skew was used as the tracking

signal by the PI controller and the output of this controller was used to discipline the clock by modulating the clock rate to match that of the leader. Experimental results demonstrate that this technique, which used a Kalman filter and compensated for skew, performs better than the prior technique that used a conventional digital filter and compensated for offset. These results demonstrate that this technique is resilient against packet losses and achieves better accuracy and stability for both single-hop and multi-hop scenarios. In the future, this technique can serve as a foundation to improve determinism and predictability in networked embedded systems.

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor, Dr. Shivakumar Sastry, for his support, guidance, and insights that made this thesis possible. My gratitude also goes to my committee members, Dr. Nathan Ida and Dr. James Grover, for their understanding and important comments. I am very grateful to Dr. Alex De Abreu Garcia and Dr. Okechukwu C. Ugweje who made it possible for me to pursue a master's degree in the department. I want to express my sincere thanks to "the boss", Gay Boden, for making life ever smoother, and Eric Rinaldo, for his patience and being a great resource. My gratitude also goes to my lab colleagues, Kranthi Mamidisetty, Branden Archer, John McGonnell, and Maithili Ghamande, for their support and useful ideas.

My friend, Francis Tetteh, derserves my sincere thanks as well for always being there when I needed help. Finally, I dedicate this thesis to my family who kept me motivated with their love and endurance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Networked embedded systems, which rely on inexpensive, resource-constrained nodes and interact in a peer-to-peer manner over wireless links, offer new opportunities for systems architecture and design in a variety of domains. Such nodes are based on inexpensive hardware, and transceivers that have relatively low bandwidth and high packet-loss rates. For example, a *relative skew* of 3000 parts per million (ppm) was measured between two Mica2Dot motes; this represents an *offset* of about 10.8 seconds over an hour. To limit the packet-loss rate on these nodes to around 10%, the nodes had to be operated at a maximum bandwidth of 19.2 kbps with Manchester encoding. Consequently, because of the resource constraints, it is challenging to synchronize activities on multiple nodes. Nevertheless, time synchronization across asynchronous nodes is an important foundation for engineered systems [1, 2].

Among the synchronization techniques reported in the literature, the IEEE 1588 Precision Time Protocol (PTP) standard is widely used in a variety of applications over wired networks [3]. To effectively apply PTP over wireless links, it is necessary to compensate for (a) the skew between the nodes, and (b) the packet-loss. In addition, processor interrupts and other hardware and software jitter in the nodes give rise to intermittent offset noise spikes in a software only implementation [4].

Filtering is a commonly used technique to eliminate noise spikes. While a conventional digital filter can attenuate such spikes [4] to some extent, a Kalman filter [5] provides better attenuation. In addition, a Kalman filter has (a) low mean error in tracking a Gaussian-distributed signal, (b) lower computational overhead when the distribution is stationary, and (c) improved robustness in the presence of lost signals. For these reasons, a Kalman filter was selected as a pre-processor.

A clock servo is used to discipline the clock in nodes. The clock servo used by Branicky et al. in [4] uses offset as the input to a PI controller with a conventional digital filter as a pre-processor. This approach is susceptible to packet-losses since the offset grows linearly with consecutive message losses which can cause instability in the servo. Because clock drift in a node is relatively constant over short durations, a Kalman filter was designed to track the skew, which is the rate of change of offset. The skew was then used as input to a PI controller in the clock servo. The use of skew mitigated the effects of packet-loss and attenuated large spikes in the offset.

However, it was necessary to derive the closed loop system function for this control model and establish its stability. The results show that this approach achieves better synchronization accuracy and reduces Allan variance in all time scales.

1.1 Contributions

The contributions of this thesis are:

1. The design of an energy-efficient, flexible, and multicast-capable Media Access Control (MAC) protocol that is based on enhancements to the B-MAC protocol.
2. Implementation of this MAC protocol on the Mica2Dot wireless sensor node platform.
3. The design of a clock servo that tracks clock skew and uses a discrete linear Kalman filter as a pre-processor.
4. Implementation of an IEEE 1588 style time synchronization on the Mica2Dot platform using the above clock servo.

1.2 Overview of this Thesis

Following a review of related work in Chapter 2, Chapter 3 presents the MAC protocol used in this investigation. Chapter 4 presents the design and implementation of a IEEE 1588 style synchronization technique on the Mica2Dot motes. Chapter 5 presents experimental results. The conclusion and next steps are discussed in Chapter 6.

CHAPTER II

BACKGROUND

Establishing and maintaining a common notion of time across asynchronous, distributed, nodes that are not physically connected requires communication of messages with *implicit* or *explicit timestamps* [6]. In distributed systems that share a communication channel, a *media access control* (MAC) protocol is often used to enable this sharing [7]. This chapter reviews MAC protocols for networked wireless embedded nodes with constraints of energy, bandwidth, computation, and memory capacity. The chapter also presents related work in time synchronization and a brief introduction to the IEEE 1588 PTP.

2.1 MAC Protocols

Wireless embedded nodes have limited energy, bandwith, computational, and memory capacity. Several MAC protocols have been proposed to address this unique challenge [8].

Duty cycling is a well-known technique of conserving energy. In this scheme, nodes periodically cycle between a sleep period and a wake period. This conserves energy by avoiding idle listening, which wastes a bulk of the energy in most applications [9]. If the sleep/wake schedule of a target node is unknown, then a sending

node must transmit a "dummy"' message, referred to as a *preamble*, for the duration of the sleep/wake schedule before transmitting the actual message. This guarantees a *rendezvous* with the receiver, i.e., ensures the receiver is awake to receive the message. This method of communication is referred to as *preamble sampling*.

The B-MAC protocol proposed by Hill et al. [10] is a *carrier sense multiple access* (CSMA) protocol that achieves low power operation through duty cycling with preamble sampling. The nodes use a fixed duty cycle but they do not know the sleep/wake schedule of their neighbors. Channel assessment for CSMA operation is performed by first re-estimating the noise-floor after transmitting every packet and then using this as a threshold to determine channel state before the next transmission. B-MAC provides basic functionality but includes bidirectional interfaces that can be used to obtain extended functionality. However, a major drawback of B-MAC is the potential waste of bandwidth and energy by the long preamble that precedes every message.

S-MAC [11] is a contention-based MAC protocol that also uses fixed duty cycling to minimize energy waste. However, unlike B-MAC, nodes must broadcast their sleep/wake schedule to their neighbors. This avoids the need for preamble sampling but introduces the need for time synchronization. To ensure that nodes in a local area adopt the same schedule, a new node in the network must first listen for a schedule from its neighbors and adopt all the schedules it hears. Nodes that are already in the network must add any new schedule they hear. This coordinated sleeping may lead to increased collisions as nodes wake at the same time and try to

transmit immediately. S-MAC also incurs a high memory overhead from the need to store multiple sleep schedules.

T-MAC [12] further reduces idle listening in S-MAC by setting a time out for listening. When a node wakes up, it listens for a short duration of time and goes back to sleep early if it does not detect any transmission on the channel. T-MAC suffers the same high probability of collisions and memory overheads as S-MAC.

2.2 Time Synchronization

Time synchronization on wireless embedded nodes that have constraints of energy, bandwidth, computation, and memory capacity remains a challenge because any resource used for synchronization may significantly reduce the resources available to perform the network's fundamental task [13]. Eventhough several protocols for time synchronization on such nodes have been proposed in the literature [14, 15, 6, 16, 17], a few relevant ones are reviewed below.

In *Reference Broadcast Synchronization* (RBS) protocol [15], a beacon node broadcasts a reference message to nodes in its neighborhood. The nodes then exchange the time each of them received the message among themselves. After receiving several messages and using linear regression, each node is able to compute its offset and skew from the other nodes. Consequently, a node can estimate the time on another node through timescale transformations. An important feature of RBS is that it does not depend on a single clock for time reference. Nodes are synchronized pairwise. However, the pairwise exchange of several messages represent a significant

communication overhead and therefore makes RBS unscalable. Also, with the linear regression, nodes cannot act quickly to changes in skew and offset since they must wait for several messages before they can re-estimate these parameters.

In [16], Maróti et al. proposed the *Flooding Time Synchronization Protocol* (FTSP). FTSP organizes the nodes into a hierarchy with the root node designated as the leader. The leader periodically broadcasts a synchronization message that contains its time. This time is recorded by the receiving nodes and then re-broadcast to nodes down the hierarchy after updating the timestamp. After receiving eight synchronization messages, a node uses linear regression to compute its offset and skew from the leader. FTSP synchronization precision is likely to significantly degrade in multi-hop scenarios for two reasons: first, it does not account for the one-way delay in the communication path, and second, nodes do not act quickly to changes in clock rate and offset as they have to wait eight synchronization periods to re-estimate these parameters. Since linear regression methods are affected by outliers because the data is weighed by the square, RBS and FTSP are vulnerable to outliers in offset.

Loschmidt et al. [18] present a hyperbolic navigation algorithm for localizing 802.11 client nodes by using base stations that were synchronized by means of IEEE 1588. They used a digitally processed analog signal from the transceiver as a trigger to timestamp received messages.

Eidson et al. [19] report the jitter of three pairs of timestamp trigger signals in the MAC-PHY interface of a Cisco AIRONET series 340 IEEE 802.11b WLAN card. Their results show the lowest jitter with a standard deviation of 145.6ns can

be achieved when the rising edge of TX_RDY (transceiver ready to receive a packet from processor) and MD_RDY (transceiver ready to transfer a packet to processor) are used. This work shows that it is important to obtain timestamps, using appropriate triggers, at the hardware level.

The software-only design and implementation of the IEEE 1588 on ethernet by Branicky et al. in [4] is closely related to this work. Branicky et al. [4] presented the results achieved by an application-layer implementation of a PTP daemon on nodes executing under the Linux operating system. To isolate clock read jitter caused by interrupts, they used a hardware clock on the node. Their analysis used the notion of small, medium, and large time-scales for clock stability using *Allan variance* as a metric. While the Allan variance of the disciplined clock was decisively less than that of the undisciplined clock in the large time scale (over 100 seconds), and fairly close to the undisciplined clock in the small time scale (less than a few seconds), this variance was large for the medium time scale (10 to 100 seconds). Their filter did not prevent noise spikes in the offset from entering the PI controller. We believe that consecutive message losses can produce a spike in offset and exacerbate the medium time scale instability. The servo design is therefore susceptible to message losses and thus not well-suited for the wireless environment.

2.3 The IEEE 1588 Synchronization Protocol

IEEE 1588 PTP is a time synchronization protocol for distributed systems. It is an *external synchronization protocol* in which all clocks in the network trace their time

to a single clock known as the *grandmaster clock*. The clock that PTP uses to derive its time is referred to as the *leader* clock and the clock that derives its time from a leader clock is called a *follower* clock.

PTP achieves time synchronization in two steps: First, the clocks in the network are organized into a leader-follower hierarchy, and then timestamped messages are exchanged between leader and follower clocks to synchronize them. At the start of the protocol, the clocks broadcast messages containing information about their "quality". Using this information and a distributed spanning tree algorithm referred to as the *Best Master Clock* (BMC) algorithm, each clock independently determines the best clock in its broadcast domain. The best clock becomes the leader in the broadcast domain and the rest become follower clocks. The BMC algorithm logically partitions the network into a clear leader-follower hierarchy of clocks.

Once this clock hierarchy is formed, the nodes exchange four types of timestamped messages to establish and maintain synchronization. These messages are *Sync*, *Follow_Up*, *Delay_Req(uest)*, and *Delay_Resp(onse)* messages. Sync and Follow_Up messages are transmitted by a leader clock to allow a follower clock to calculate its offset. The Follow_Up message only transports the timestamp of the preceding Sync message. A Delay_Req message is transmitted by a follower clock and the leader replies to it with a Delay_Resp message containing the time the Delay_Req message was received. This exchange allows the follower to estimate the *one-way delay* of the communication path. Suppose a leader transmits a Sync message at time T_1 and it it is received by the follower at time T_2, according to its local clock. Also, suppose

a Delay_Req message is transmitted by the follower at time T_3 and it is received by the leader at time T_4. If the offset of the follower clock is $offset\_from\_leader$, and the one-way delay of the communication path is one_way_delay and symmetric, then these two parameters can be calculated as:

$$one\_way\_delay = \{(T_2 - T_1) + (T_4 - T_3)\}/2 \tag{2.1}$$

$$offset\_from\_leader = (T_2 - T_1) - one\_way\_delay \tag{2.2}$$

Incorporating the one-way delay in the calculation of the offset is critical for high precision synchronization. How this offset is used to appropriately correct, or discipline, the clock is not specified by the protocol.

CHAPTER III

MEDIA ACCESS CONTROL PROTOCOL

In networked embedded systems with asynchronous nodes communicating over a shared communication medium, it is critical to arbitrate access to the medium. A Media Access Control (MAC) protocol is used for such arbitration [7] in wired and wireless networks. The MAC was particularly important for this investigation into inband, software only, IEEE 1588 style synchronization over wireless links because the temporal properties of the MAC impact the synchronization accuracy across nodes in the system. For example, a poorly designed MAC can increase non-determinism in message latency [14], and excessive delays between the IEEE 1588 *synchronization* and *follow-up* messages can destabilize the control loop that is used to discipline the clock [3].

This chapter presents the design of the MAC protocol that was implemented to support this investigation. The well-known B-MAC (Berkeley-MAC) protocol [10] was enhanced to improve latency, throughput and flexibility. This design was implemented on the Mica2Dot platform that uses an Atmega128L microcontroller – an 8-bit processor running at 7.37 MHz – with an integrated CC1000 radio transceiver which communicates with the processor over an SPI interface [20]. After briefly discussing

the design objectives and the enhancements, critical issues that were important for the implementation of the CC1000 driver are presented.

3.1 Design Objectives

Network parameters and Quality of Service (QoS) dictated by application requirements guide the choice of a MAC protocol. For example, if throughput is the QoS of interest, then TDMA protocols are generally better suited for high node degree and high traffic scenarios while CSMA protocols perform better in low traffic scenarios. Thus, flexibility, i.e., the ability to morph the behavior of the MAC protocol to meet application requiremets, is important.

The resource-constrained environment of the Mica2Dot platform motivated us to focus on a light-weight design that could be implemented in a simple manner. The Mica2Dot platform has only 128K bytes of program memory, 4K bytes of data memory, and a maximum radio bandwidth of 38.4 Kbps. The CC100 radio does not provide any buffer to support the communication. A MAC protocol on such platforms must therefore have low computational, memory, and bandwidth overhead. Further, it must be simple to deploy and operate nodes that utilize the default behavior of this MAC protocol. Thus, the three primary design objectives were

- flexibility,
- simplicity, and
- energy-efficiency.

As disucussed in Chapter 2, there are a variety of MAC protocols reported in the literature, each with its own objectives and application. Consequently, the design of yet another MAC protocol was not a primary objective of this investigation. Among these protocols, the B-MAC protocol [10] had many interesting attributes that met the above design objectives in the context of this investigation. The following sections describe the design and implementation of the MAC protocol in detail.

3.2 Design of the MAC Protocol

The main attraction of B-MAC was the design structure that allowed users to modify its behavior through a programming interface (API) to support a variety of behaviors. However, B-MAC transmits a long preamble before, and estimates the noise-floor after, sending every message. The radio is always put to sleep after a transmission or reception and no API is provided to disable/enable this behavior. To utilize B-MAC for this investigation into time synchronization, it was necessary to remedy these shortfalls. The enhancements are presented following this brief review of B-MAC.

3.2.1 B-MAC Protocol Design

B-MAC is a light-weight CSMA protocol for resource constrained nodes that uses *duty-cycling* with *preamble sampling* to achieve low energy operation. Nodes periodically cycle between sleep and wake states. If an alternating sequence of zeros and ones, called a *preamble*, is detected in the wake state, the node stays awake to receive the data that is probably being sent to it by another node in the neighborhood. Oth-

erwise, the node goes back to sleep. This periodic sleep/wake schedule is known as *duty-cycling* or *low power listening* [10].

To ensure the destination node is listening when a message is transmitted, a source node precedes every message with a long preamble, at least as long as the interval between consecutive checks of the channel by the nodes – the *check interval*. This is known as *preamble sampling*. Figure 3.1 illustrates the need for the long preamble in low power listening communication. In this figure, both the sender and receiver have the same pre-defined sleep/wake duty cycle but different sleep/wake schedules. By its schedule, the receiver wakes up to listen for a preamble at t_1 and t_5, i.e, the check interval is 4 units of time. The sender initiates a message transmission at t_2 just after the receiver goes to sleep. Note that the sender must transmit the preamble for 3 units before the receiver wakes up again at t_5. Then the receiver must stay awake for about 1.5 units until the preamble is completed before it can receive the data from the sender.

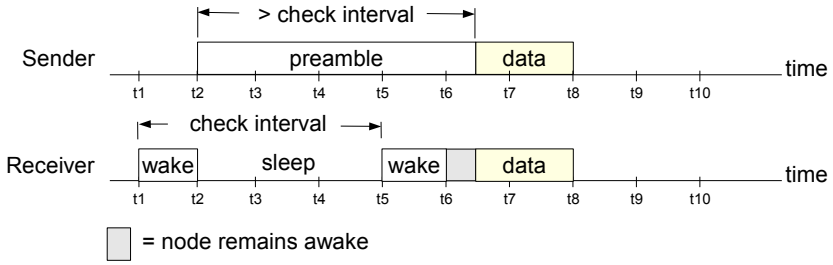


Figure 3.1: A sending node transmits a preamble for at least the duration of the *check interval*. This ensures that the receiving node is awake when the actual data is transmitted. The check interval value is pre-defined in all the nodes.

Conservation of energy from the low power listening strategy in B-MAC is based on the assumption that nodes spend more time listening than transmitting. Thus, this approach shifts the energy burden from receivers to senders by making senders consume a little more energy with the transmission of the long preamble. Analytical and experimental data presented in [10] clearly shows that, given a network topology and traffic conditions, there is an optimum value for the check interval: if it is too small, energy is wasted by idle listening, and if it is too large, energy and bandwidth are wasted by the long preamble.

The design strategy in B-MAC is to provide basic functionality and leave any extended functionality to upper layers. For example, it does not include channel reservations or service differentiation schemes. It therefore incurs minimal computational and communication overhead. Additionally, with the preamble sampling, a node does not need to store the sleep schedule of its neighbors as required in the S-MAC [11] and T-MAC [12] protocols. This enables B-MAC to have very low memory overhead. Also, the CSMA operation makes it easy to deploy and operate.

Clear channel assessment (CCA) is critical for CSMA operation and is the central idea in B-MAC. The CCA is executed in two parts. First, the noise-floor is estimated. This estimate is then used to assess the state of the channel, i.e., whether the channel is clear for sending a message. The noise floor is estimated by collecting n_f Radio Signal Strength Indication (RSSI) samples from the radio when the channel

is idle[1]. The median of the samples is used to update an exponential running average with decay factor α. This average is the noise-floor estimate. Prior to sending a message, every sending node must assess the state of the channel by collecting n_c RSSI samples. If the value of any of these samples is below the noise-floor estimate, the channel is considered to be idle. This technique is referred to as *outlier detection*. If all the n_c samples are above the noise-floor estimate, the channel is considered to be busy and the node will not initiate a transmission. Collisions are avoided or resolved by randomly backing off before a transmission. The suggested values for these parameters are $n_f = 10$, $\alpha = 0.06$ and $n_c = 5$ for a typical wireless sensor environment [10].

To facilitate reliable communication B-MAC supports acknowledments (ACK). Figure 3.2 shows a typical sequence of messages exchanged between a sending and receiving node when acknowledgments are enabled. After receiving a message, the receiving node immediately transmits an ACK message without clear channel assessment and the long preamble. The sending node waits for the ACK and times out if it is not received.

Figure 3.2 also shows that B-MAC does not prevent overhearing. All the neighbors overhear messages that are sent to a node in the vicinity. While this multicast behavior may not be energy efficient, it is a requirement for the IEEE 1588 style synchronization investigation.

[1]For example, immediately after a transmission or reception, or when valid packets are not being received

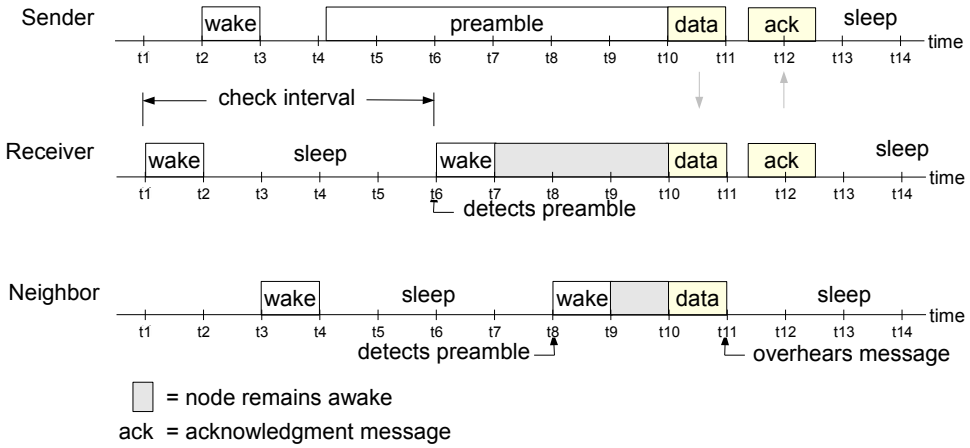


Figure 3.2: Sequence of messages exchanged in reliable communication mode. Acknowledgements are sent immediately without clear channel assessment and the long preamble.

B-MAC provides a flexible design. Various behaviors of the MAC can be obtained by setting appropriate parameters. For example, to implement a scheduling-based MAC, clear channel assessment may be disabled, the preamble length set to the minimum required for radio synchronization, and check interval set to occur in the receive timeslots. Other MAC protocols reported in the literature such as S-MAC [11] and ZMAC [21] have also been realized using B-MAC. The B-MAC API provides flexibility and allows for changes in the default behavior of the protocol. Table 3.1 shows the methods in the three sets of interfaces defined. *MacControl* provides interfaces to enable/disable acknowledgements and CCA. Methods in the *low power listening* set allow users to change tranmit/receive power, check interval and length of the preamble. *MacBackoff* methods set the initial and congestion backoff windows.

Table 3.1: B-MAC bidirectional interfaces

Interface	**Methods**	**Purpose**
MacControl	EnableCCA	enables channel sensing
	DisableCCA	disables channel sensing
	EnableAck	enables message acknowledgements
	DisableAck	disables meassage acknowledgements
MacBackoff	InitialBackoff	backoff window size for first tx attempt
	CongestionBackoff	backoff window size if tx fails
LowPowerListening	SetListeningMode	sets radio to low/high power rx mode
	GetListeningMode	gets the radio rx power mode
	SetTransmitMode	sets transmit power
	GetTransmitMode	gets transmit power
	SetPreambleLength	sets length of preamble
	GetPreambleLength	gets length of preamble
	SetCheckInterval	sets channel check period
	GetCheckInterval	gets channel check period

Despite the many positive attributes of B-MAC, there are some shortfalls that make it inefficient for networked embedded systems. First, consider a scenario in which a node needs to transmit multiple messages to the same receiver. Preceding each message with the long preamble is unneccessary, and wastes energy and bandwidth. It is more efficient to transmit the long preamble only once with the first message. Second, since the default API does not support methods to enable/disable the radio sleep, it was necessary to add such methods to the MacControl interface set. For nodes that are energy constrained, it is useful to put the radio to sleep but for nodes that have no energy constraints, it is harmful to put the radio to sleep as demonstrated in the next section. Finally, it may not be necessary to frequently re-estimate the noise-floor in several application domains. Based on these observations, the following improvements were made to remedy the shortfalls.

3.2.2 Improvements to B-MAC

The three improvements designed to enhance B-MAC were:

1. Burst Mode Signalling,
2. Radio Sleep Enabling/Disabling, and
3. Effective Clear Channel Assessment.

Burst Mode Signalling

Instead of transmitting long preambles before every message, it was more efficient to design a signalling scheme to notify the receiver that it must continue to remain awake to receive further messages.

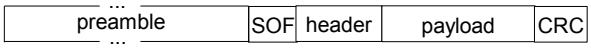
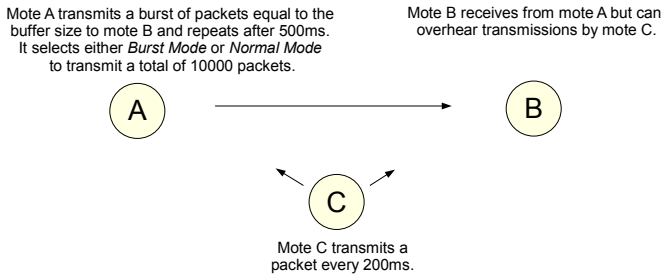


Figure 3.3: Typical packet format in a preamble sampling regime. A SOF byte is used to indicate the beginning of the message packet.

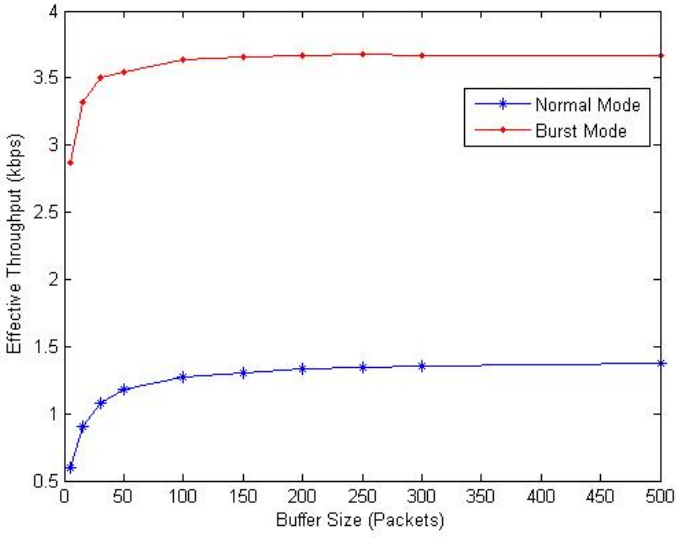
Figure 3.3 shows the typical structure of a packet in communication systems that require a preamble. The *start-of-frame* (SOF) byte is used to indicate the beginning of the message packet. By allowing the SOF byte to take one of two values, the sending node could signal the receiving node whether this message is the first in a sequence or not. The overhead for this signalling mechanism is minimal — only a single logical comparison. We refer to the exchange of messages with this signalling mechanism as *burst mode transmission*. While the burst mode does not guarantee per-packet fairness across the nodes, such fairness is not important for many applications. For example, the IEEE 1588 protocol specifies a *Burst Mode Operation* in which the *Sync* and *Delay_Req* messages are transmitted more frequently to quickly synchronize clocks at startup [3].

Figure 3.4(b) compares throughput performance of burst mode and normal mode that was experimentally obtained using the setup in Figure 3.4(a). The effective

throughput was calculated by dividing the total number of packets correctly received by the duration of transmission. The higher throughput in burst mode may be attributed to the reduced waste of bandwidth by the preamble, as well as, reduced packet losses due to guaranteed rendezvous.



(a) Experimental setup



(b) Throughput performance

Figure 3.4: Effective throughput is significantly better using *Burst mode*. Gains in throughput with buffer size, however, flattens out after 100 packets in both modes. The check interval and preamble length were set at 20ms and 24ms, respectively.

Enabling/Disabling Radio Sleep

The CC1000 radio required up to 6ms to wake up from a sleep cycle [22] [2]. This delay in the wake time results in reduced throughput, availability, and/or increased packet-loss rate as demonstrated in Figure 3.5. This figure shows results from an experiment in which one mote sent 10,000 packets to another mote. A packet was transmitted every 200ms. This was repeated for different values of the check interval. The packet-loss rate is smaller at all check intervals with sleeping disabled because the delay in waking the radio increases the likelihood of the receiver not being in receive state when a packet is transmitted. The high packet-loss rate for small values of the check interval may be attributed to interrupt latency causing a rendezvous to be missed. The increasing packet loss rate for large values of the check interval is likely caused by an increased probability of missing a rendezvous due to the fact that the increased listening timeout may not adequately compensate for oscillator drift. This result demonstrates the need for enabling/disabling radio sleep.

Effective Clear Channel Assessment

Effective clear channel assessment is important for CSMA protocols because underestimating the noise-floor may lead to unnecessary backoffs while over-estimating it may cause collisions. Both increase message latency and negatively impact synchronization precision.

[2]While periodic sleep/wake operation is useful for energy constrained nodes, this is not important for nodes that are powered from conventional sources. For example, nodes in a factory environment can be powered from a wall outlet.

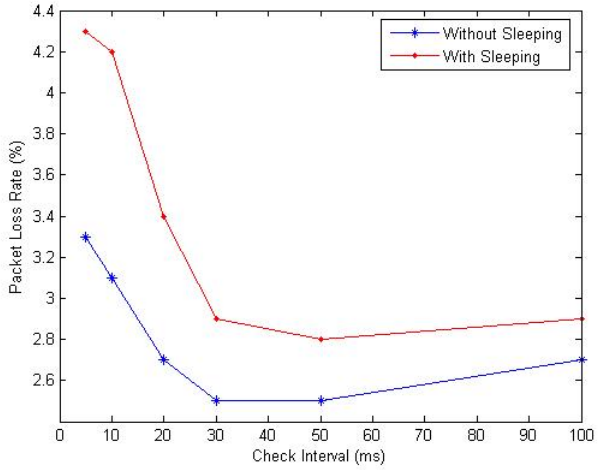


Figure 3.5: Packet-loss rate is better when radio sleeping is disabled. In either case, there is an optimum check interval for the lowest packet-loss rate. Preamble length was made 10% longer than the check interval to compensate for increased phase drift at higher check intervals.

CCA schemes that use signal strength as the decision criterion operate on the assumption that the average signal energy during a transmission is always greater than the noise-floor. This assumption was empirically found to hold true for the CC1000 transceiver. The signal strength was measured by a mote at three locations in the wireless environment while another mote was transmitting a packet every two seconds. These three regions were: (1) well within the communication range, (2) at the edge of the communication region[3], and (3) outside the communication region. As can be seen in Figure 3.6, the received signal strength is well above the noise-floor even at the edge of the communication range. These results indicate that much of

[3]Locations where a preamble may be detected but a packet is never received correctly.

the error associated with channel state determination can be avoided by using the maximum instead of the median RSSI value to estimate the noise-floor.

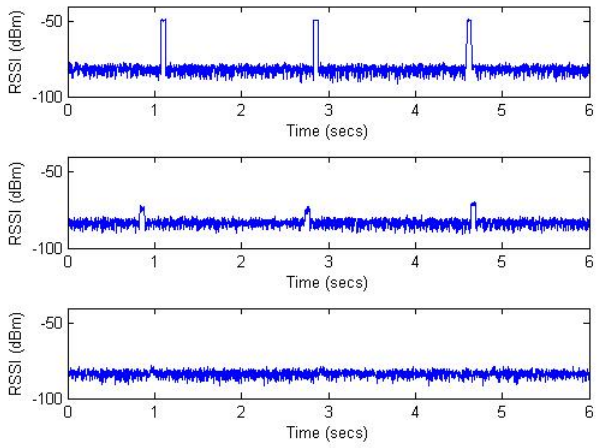


Figure 3.6: RSSI measured by a receiver in communication range (top), at edge of communication range (middle), and out of communication range (bottom) of a transmitter.

In the CCA scheme used for this investigation, 11 RSSI samples were collected when the channel is known to be idle – e.g. when no valid message is received in two check intervals. The maximum value in the samples is used as an estimate of the noise-floor which serves as threshold to determine channel state. The noise-floor may only need to be re-estimated after excessive retransmission failures or backoffs. To determine the state of the channel, 5 RSSI samples are taken and if the minimum value is less than the noise-floor, the channel is considered idle. Otherwise, the channel is considered busy. Figure 3.7 shows that this scheme correctly detects the channel state at all times compared with B-MAC's scheme that frequently detects the channel busy when it is actually idle.

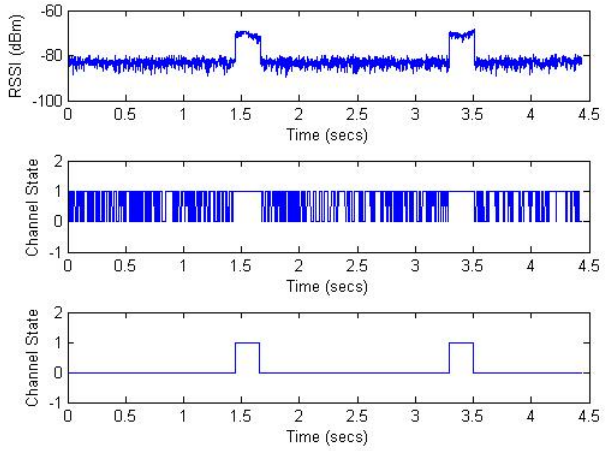


Figure 3.7: RSSI measurements were taken every 2ms while a node was periodically transmitting a message every 2secs (Top). Our method (bottom) correctly detects the channel state at all times compared with BMAC's method (middle) that frequently detects the channel busy when it is actually idle.

It must however be noted that the use of RSSI for channel assessment may become ineffective for radio transceivers that use spread spectrum techniques since a message may still be correctly received even if the signal energy is below the noise-floor [23]. In such tranceivers, detection or non-detection of a valid spreading code may be used as the criterion to determine channel state.

3.3 Implementation

The Chipcon CC1000 radio integrated on the Mica2Dot platform uses *binary frequency shift keying* (BFSK) modulation and has no buffer. This modulation scheme does not provide any robustness against noise encountered during physical transport of data. The enhanced B-MAC protocol discussed in the preceding section was

implemented in the nodes and only supported a CRC-based error detection for the packets. The protocol was implemented in two parts, namely the cc1000 driver and the B-MAC state machine.

To isolate operating system issues from media access control and synchronization issues, a simple node-level operating system was used.

3.3.1 Basic Operating System Structure

A non-preemptive FIFO scheduler with two levels of priority was implemented using two queues and a dispatcher. The dispatcher removes the task at the head of the higher priority queue for execution. It dispatches tasks in the lower priority queue only if the higher priority queue is empty. The higher priority level was assigned to tasks posted from within an interrupt service routine.

A system software clock was implemented using COUNTER1 on the Atmega128L. A clock update rate of once per millisecond was used as a compromise between timer interrupt processing overhead and clock resolution. Consistent with the IEEE 1588 standard, time was represented as an 8-byte structure. The first 4 bytes of type integer stored the number of seconds and the second 4 bytes of type unsigned integer stored the number of nanoseconds from a pre-defined epoch.

In addition to the software clock, B-MAC and the time synchronization scheme discussed in Chapter 4 required timers to operate. These timers were also derived from COUNTER1. On every "tick of the clock", a millisecond counter was incremented and the timer data was checked to see if any of them had expired. This

added more processing overhead in the timer interrupt and care was taken to ensure that the service time of this interrupt did not exceed the duration required to transmit or receive a byte for reasons explained in the next section. To ensure this constraint was not violated, the number of timers was limited to five.

3.3.2 CC1000 Driver

Because of the simple modulation scheme (BFSK) and lack of buffer in the CC1000 transceiver, it was necessary to design and implement optimization strategies into the driver to reduce packet errors. The lack of buffer in the transceiver meant that the microcontroller was required to participate in the sending of every bit of data. The SPI interface in the Atmega128L has a single 8-bit transmit buffer but a double 8-bit buffer for receive. Consequently, depending on whether polling or interrupt is used for transmission, interrupt service times on the microcontroller may not exceed the duration of time it takes to transmit a bit or a byte by the transceiver.

For optimum receiver sensitivity, the radio was configured to operate with Manchester coding with maximum frequency separation of 64kHz between bit 0 and bit 1, and at recommended subcarrier frequencies using the SmartRF Studio program provided by Chipcon. The oscillator and PLL currents were also programmed high by default.

Microcontroller-Radio Interface

Figure 3.8 depicts the interface between the Atmega128L and the CC1000 on the Mica2Dot platform.

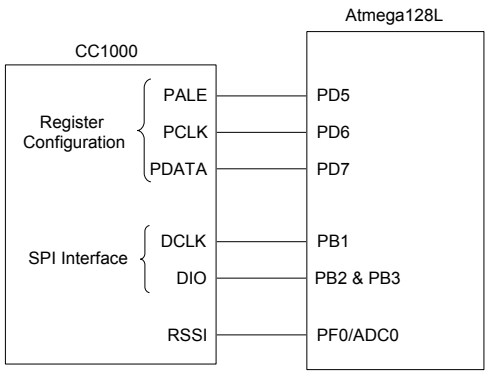


Figure 3.8: Atmega128L-CC1000 interface on the Mica2Dot mote.

The PDATA, PCLK, and PALE lines serve as serial interface for configuring the radio through *bit-banging*. To write to any of the 8-bit configuration registers on the CC1000, a 2-byte configuration word is placed serially on the bidirectional PDATA-line beginning with the most significant bit (MSB). The bits are transferred on the negative edge of a software-generated PCLK signal. The most significant seven bits of the configuration word represent the register address, the next bit is the mode bit, which indicates whether a read or write operation is being performed, and finally the last eight bits is the value to be written to the register. The PALE is held low during the address and read/write byte and changed to high during the transfer of the register value. The same procedure is used for a read operation except data flows from the CC1000 to the microcontroller on the PDATA-line and a read operation must be specified in the mode bit. The CC1000 datasheet [22] contains details about timing diagrams and the registers. The pins DIO and DCLK are *serial peripheral interface* (SPI) pins used to transfer data between the microcontroller and CC1000 which acts

as the SPI master. The RSSI/IF pin provides an analog signal that represents the received signal strength. Data may be received from the radio using two methods: polling or interrupt. Polling was favored for three reasons. First, the SPI clock provided by the CC1000 is continuously running and will generate an interrupt every 8 clock cycles if the SPI interrupt is enabled. This may represent a significant processing overhead if the interrupt service time is not negligible. Second, polling fits naturally with our duty cycling MAC protocol. Third, interrupts increase message delay non-determinism and thus affects synchronization accuracy. However, using pure polling for transmit meant that to avoid causing frame errors the microcontroller could not be interrupted for more than a bit duration during transmit. This was considered too tight a constraint to guarantee its non-violation. Hence a hybrid approach was adopted where interrupt transmit was used but the SPI interrupt was only enabled during a transmission.

Message Handling

Every frame starts with a specified number of preamble bytes (0x55 or 0xAA). The preamble was used to rendezvous with the receiver node, i.e., ensure that the receiver is receiving when the sending node is transmitting. The preamble was also used to bit-synchronize the receiving node's tranceiver. The CC1000 required a minimum of 7 bytes (98 bauds) of preamble in the Manchester mode for the *averaging filter*[4] to obtain a suitable threshold value to differentiate bit 0 and bit 1. This transceiver

[4]The averaging filter measures the average voltage level of the preamble bits which is then used to detect subsequent bits. Longer settling time improves sensitivity.

synchronization requirement and the finite time required to wake the radio from sleep as well as oscillator drift are the three reasons the preamble length must be greater than the check interval. A *start-of-frame* (SOF) byte is inserted at the end of the preamble and the beginning of the frame to indicate where the frame starts in the byte stream as shown in figure 3.3.

Since a frame is transmitted as a stream of bits in the wireless medium but is received from the SPI buffer as a stream of bytes, an effective scheme is needed to parse the byte-stream on a bit-by-bit basis in order to extract the frame. Note that the received byte-stream may not necessarily be aligned as they were in the transmitter. One approach is to continuously look for the SOF byte in the last two received bytes using an 8-bit sliding window. However, this processing in the middle of a reception combined with interrupt servicing can cause the total processing time to exceed the time it takes to receive the next byte from the SPI buffer and hence loose the data. Additionally, the long preamble associated with every frame means that the processing overhead will be quite large. For these reasons, it was decided to receive all non-preamble bytes and parse it to extract the frame.

The recording of bytes began only after three consecutive non-preamble bytes. This approach has the problem that if three consecutive preamble bytes are corrupted, a frame loss will occur. However, the probability of three consecutive bytes being corrupted is low. Another constraint of this choice is that the next two bytes after the SOF byte may not be preamble bytes. Since the next two bytes that immediately follow the SOF is typically associated with an address of a node, this means that a

node cannot have an address containing 0x55 or 0xAA. The preamble byte immediately before the first non-preamble byte is always the first byte stored in the receive buffer since it may contain the first bit of the SOF.

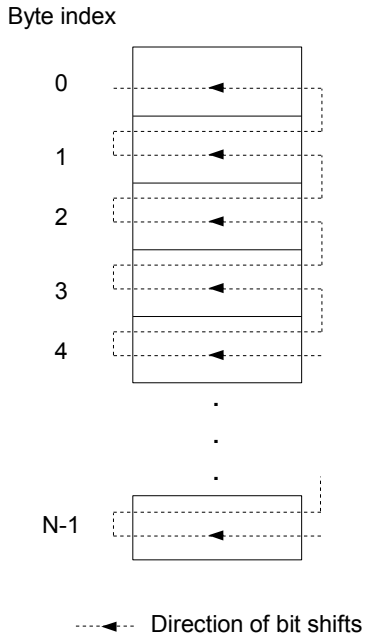


Figure 3.9: Frame parsing algorithm for an N-byte receive buffer. Bits are shifted to the left and overflow bits are placed in the least significant bit position of the preceding byte.

Once the data is received into a buffer of size N, it is parsed as depicted in Figure 3.9. Starting with the most significant bit of the first byte, bits are shifted to the left. Bits that overflow are moved into the least significant bit position of the preceding byte. This process continues until the SOF is detected in byte 0.

3.3.3 MAC State Machine

Figure 3.10 presents the state machine for the enhanced B-MAC protocol. A sleep-

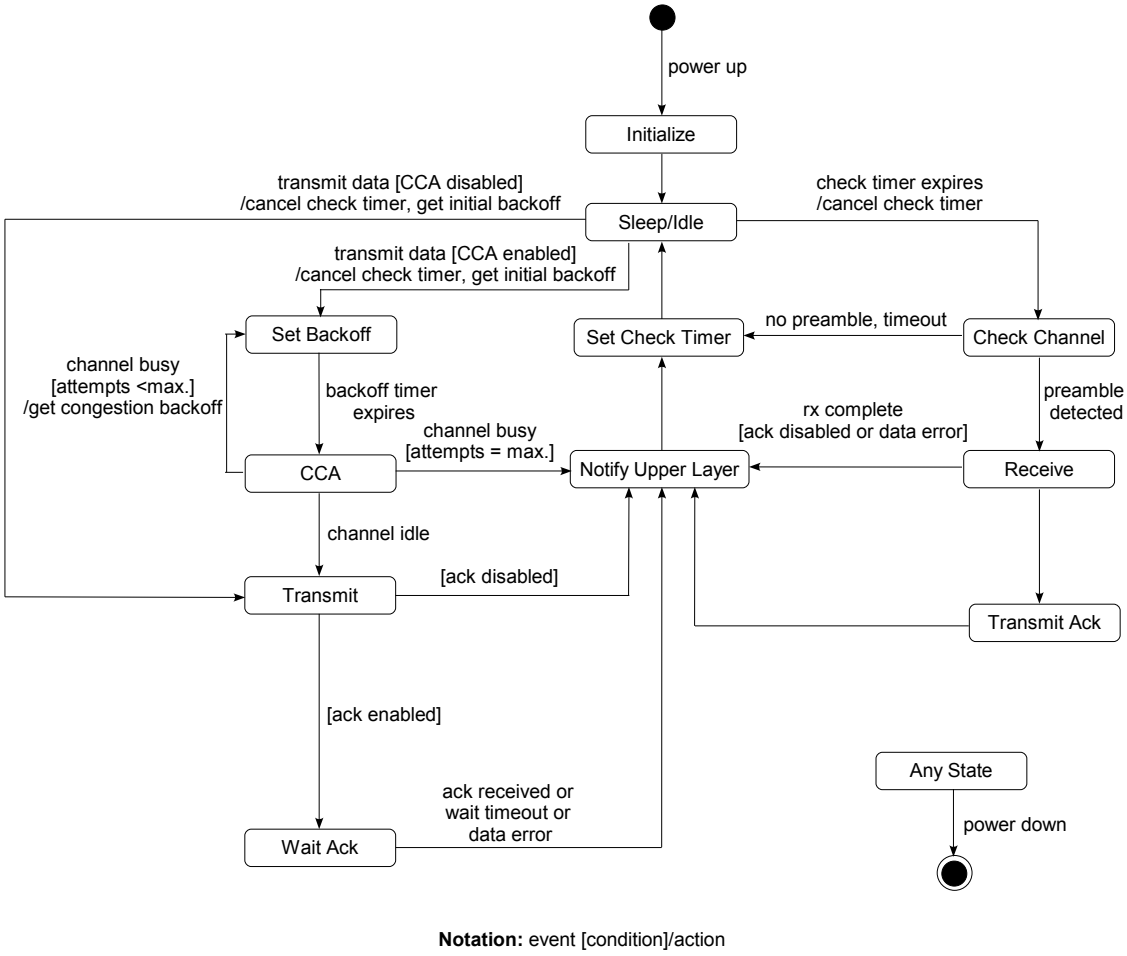


Figure 3.10: State machine for the MAC protocol.

ing or idle node wakes up periodically and checks the channel for preamble bytes and goes back to sleep if no preamble is detected after listening for 10% of the preamble duration. If it detects a preamble, it remains awake and receives the message.

If acknowledgement is enabled, the receiving node immediately transmits an ACK message if the message was received correctly, i.e., the CRC was valid. Upper layer services are notified about the reception.

A transmitting node, on the other hand, directly transitions into the *Transmit* state if CCA is disabled. Otherwise it makes a random backoff and performs CCA. If the channel is clear, it transitions into the *Transmit* state. If the channel is busy, the node backs off again. This is repeated until the channel is either clear or the maximum number of backoffs is exceeded. After transmitting, it waits for an ACK message with timeout if acknowledgement is enabled. Upper layer services are then notified about the transmission status. The noise-floor estimate is calculated once at the first opportunity and only re-estimated after excessive retransmission failures or backoffs.

CHAPTER IV

PRECISION TIME SYNCHRONIZATION PROTOCOL

This chapter presents an overview of the IEEE 1588 protocol, and the design of a clock servo used to discipline the clocks. Issues that are central to the implementation of IEEE 1588 protocol on the Mica2Dot platform are presented.

4.1 Overview of IEEE 1588

The IEEE 1588 is designed to synchronize diverse clocks with varying degree of accuracy and stability. For reliable operation and to achieve submicrosecond precision, the standard [24] requires that

1. oscillator stability and accuracy be such that a second measured by any of the clocks be within $\pm 0.01\%$ of SI second,
2. timestamps be taken as close to the hardware as possible,
3. the network must support multicast transmission,
4. communication paths be symmetric,
5. each clock be accurately described by well-defined properties.

The Mica2Dots used in the investigation do not satisfy requirement (1) as we measured skew of about 3000ppm on the motes. This may therefore limit the achievable synchronization accuracy.

The IEEE 1588 achieves time synchronization in two steps. First, a leader-follower hierarchy is established and then messages are exchanged to synchronize the clocks. These two steps are described in the next two sections.

4.1.1 Establishing Clock Hierarchy

Each clock participating in an IEEE 1588 synchronization is described by a set of properties such as its *stratum*, *identifier*, and *variance*. The stratum represents the synchronization tier of the clock to UTC by means other than PTP, the identifier represents the accuracy of synchronization to UTC, and the variance gives the stability and noise properties.

At the start of the protocol, each clock broadcasts its properties to the other clocks in its neighborhood. Using these properties about its neighbors and a distributed algorithm referred to as the *Best Master Clock* (BMC) algorithm, each clock independently determines the best clock in its neighborhood. The best clock becomes the leader while the rest of the clocks in that neighborhood become followers. The BMC algorithm results in a logical partitioning of the network into an acyclic graph and hence a clear leader-follower hierarchy. Details about this protocol can be found in [24, 25]

Two types of clocks are defined[1]: *ordinary clock* and *boundary clock*. An ordinary clock interfaces with only one broadcast domain and can serve as a leader or follower in the domain. A boundary clock interfaces with more than one broadcast domain but acts as a follower in only one of the domains, and leader in the other domains.

Figure 4.1 shows an example of a network that has been organized into a leader-follower hierarchy. Ordinary clock 1 serves as a leader clock in the network and all other clocks directly or indirectly derive their time from it, i.e, the rest of the clocks are traceable to clock 1. It is therefore the *grandleader clock*.

4.1.2 Synchronizing Clocks

Five types of messages are defined in IEEE 1588. These are *Sync*, *Follow_Up*, *Delay_Req(uest)*, *Delay_Resp(onse)*, and *Management* messages. Sync and Follow_Up messages are transmitted by the leader clock to synchronize follower clocks. A Follow_Up message is used to transport the timestamp of the preceding Sync message. Delay_Req message is transmitted by a follower clock to enable it estimate the one-way delay of the communication channel. A Delay_Resp message is transmitted by a leader clock in response to the Delay_Req message. Management messages are used by optional management consoles to configure and manage the clocks.

IEEE 1588 protocol specifies two message models for synchronization: a *one message model* and a *two message model*. Figure 4.2 shows the exchange of messages

[1]Version 2 of the IEEE 1588 protocol adds another type of clock called a *transparent clock*.

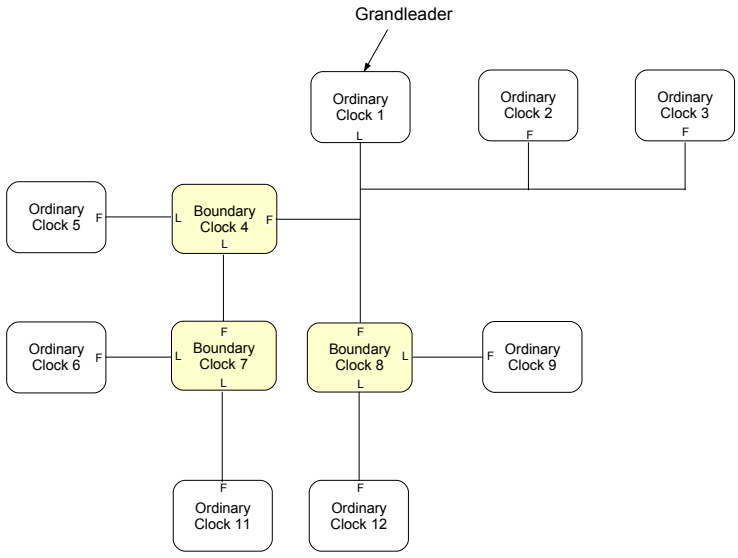


Figure 4.1: A clock hierarchy established before IEEE 1588 synchronization. Grandleader clock is an ordinary clock serving as a leader (L) in the subnet. A boundary clock serves as a follower (F) in one broadcast domain and as a leader in other broadcast domains.

in the two-message model. A Sync message is broadcast by the leader to all follower clocks in the broadcast domain every *sync period*. The time the Sync message was sent, T_1, is conveyed to the follower clocks in a Follow_Up message. When a follower clock receives the Sync message it notes the time of reception as T_2 according to its local clock. Similarly, although less frequently, a follower clock sends a Delay_Req message to its leader and notes the time of transmission, T_3. The leader responds with a Delay_Resp message that contains the time the Delay_Req message was received, T_4. The exchange of messages in the one message model is similar except that the Sync message carries its timestamp thus eliminating the need for a Follow_Up message. Suppose the follower clock has a phase offset of $offset\_from\_leader$ and the the one-

way message delay is *one_way_delay*. Then, assuming symmetric links with equal delay in both directions, the following equations calculate these two parameters:

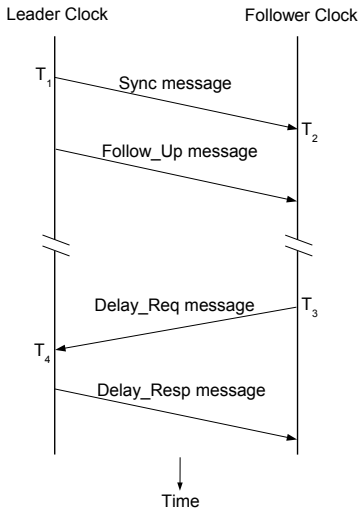


Figure 4.2: Exchange of messages in IEEE 1588 protocol's two-message model.

$$one\_way\_delay = \{(T_2 - T_1) + (T_4 - T_3)\}/2 \tag{4.1}$$

$$offset\_from\_leader = (T_2 - T_1) - one\_way\_delay \tag{4.2}$$

$(T_2 - T_1)$ is the leader-to-follower delay, *l2fDelay*, and $(T_4 - T_3)$ is the follower-to-leader delay, *f2lDelay*. Equations (4.1) and (4.2) can thus be written respectively as

$$one\_way\_delay = \{l2fDelay + f2lDelay\}/2 \tag{4.3}$$

$$offset\_from\_leader = l2fDelay - one\_way\_delay \tag{4.4}$$

If the links are highly asymmetric, the difference in message delay must be accounted for in order to maintain high synchronization precision.

4.2 Clock Servo Design

IEEE 1588 does not specify how the clocks must be disciplined. In a wireless environment where high packet-loss rate is common, loss of consecutive synchronization messages in an instantaneous clock correction regime will result in a linear increase in offset [6]. It is therefore important to aggressively estimate and correct for skew in each node using a continuous clock correction scheme. Also, a software-only implementation of IEEE 1588 faces intermittent noise spikes from interrupt jitter [4]. An effective pre-processor is therefore needed to remove these spikes.

4.2.1 Linear Kalman Filter

Several factors, such as variations in (a) medium access time, (b) propagation time, (c) radio bit synchronization accuracy, (d) delay in reading the clock, and (e) period of transceiver oscillator, as well as the limited accuracy of computations, induce jitter in timestamps in a software-only synchronization. Because oscillator jitter is a consequence of the hardware design, it cannot be corrected in software. The uncertainty in medium access time is mitigated by recording timestamps as close to the hardware as possible.

A Kalman filter was chosen to process the clock skew in order to remove the noise added by all the sources above. The *discrete linear Kalman filter* (DLKF) is

a recursive filter that minimizes the mean square error [5]. The following properties make the Kalman filter suitable as a pre-processor for the PTP clock servo:

1. The Kalman filter is an optimal estimator when the variations in the tracked signal are Gaussian-distributed. The cumulative effect of all the variations discussed above results in a Gaussian-distributed skew as shown in Chapter 5.

2. The Kalman gain effectively determines the bandwidth of the filter. This can be adjusted by choosing appropriate values of the process and measurement noise which does not change the filter delay.

3. If timestamps are recorded close to the hardware and there is no store-and-forward of synchronization messages, message delay as well as the other delays are relatively independent of traffic load. The skew distribution can therefore be assumed stationary; in this case, the Kalman gain converges to a constant value that can be pre-calculated. This reduces the filter computational overhead to one subtraction, one multiplication and one addition operation.

The temporal dynamics that govern the skew between a leader clock and a follower clock is modeled as follows. Suppose a synchronization message is sent by the leader every T seconds and the follower measures an offset of Δt_k in time step k. Let s_k represent the actual skew in time step k. Since the clock drift is assumed zero over a synchronization interval (which is usually a few seconds), then

$$s_k = s_{k-1} + v_{k-1} \tag{4.5}$$

where, v_{k-1} is *process noise*, i.e., deviations from the model due to oscillator jitter. This is a zero-mean, Gaussian-distributed random variable. Let s_k^* represent the measured skew at the node in time step k. Then,

$$s_k^* = s_k + w_k \tag{4.6}$$

where w_k represents the *measurement noise*. Because of the high packet-loss rate in the wireless environment, synchronization messages may not be received by the follower clocks regularly even if they are transmitted at regular intervals. T may therefore vary from time step to time step from the perspective of the follower.

Equation (4.5) and Equation (4.6) represent the standard model for the DLKF. Let

- Q and R represent the variance of v and w, respectively,
- $\hat{s}_k$ represent the estimated skew in time step k,
- $\hat{s}_k^-$ be the predicted estimate of the skew,
- P_k^- be the predicted error variance,
- K_k represents the Kalman gain, and
- P_k the corrected error variance.

Then the Kalman filter equations we use are:

Prediction:

$$\hat{s}_k^- = \hat{s}_{k-1} \tag{4.7}$$

$$P_k^- = P_{k-1} + Q \tag{4.8}$$

Correction:

$$K_k = P_k^- (P_k^- + R)^{-1} \tag{4.9}$$

$$\hat{s}_k = \hat{s}_k^- + K_k(s_k^* - \hat{s}_k^-) \tag{4.10}$$

$$P_k = (1 - K_k)P_k^- \tag{4.11}$$

The skew estimated by this filter is used as input to the PTP clock servo as explained in the next section.

4.2.2 PTP Clock Servo

A PI controller is commonly used to discipline clocks [3, 4]. For a specified synchronization accuracy, such a controller allows one to have a longer synchronization interval, i.e., the duration of time between two successive synchronization messages. This frees up communication bandwidth for application messages, instead of synchronization messages.

PI controllers reported in [3, 4] adjust the clock tick rate, i.e., the oscillator frequency or model parameters of a software clock, in response to the offset that is being tracked. This approach is susceptible to packet-losses. For example, if the leader's clock rate is not closely tracked or the follower clock has poor holdover properties, loss of consecutive Sync messages produce spikes in the offset and lead to instability in the PI controller [4]. It is therefore critical to closely track the leader clock rate in a lossy communication environment. Therefore, the PI controller was designed to adjust the clock rate in response to changes in the clock skew. Two immediate benefits of this approach are:

1. it is insensitive to message losses because clock skew remains practically constant over a few synchronization intervals,

2. it attenuates noise in the offset since the skew is estimated by dividing offset difference, which is in microseconds, by the synchronization period, which is in seconds.

Figure 4.3 shows a control block diagram of the clock servo.

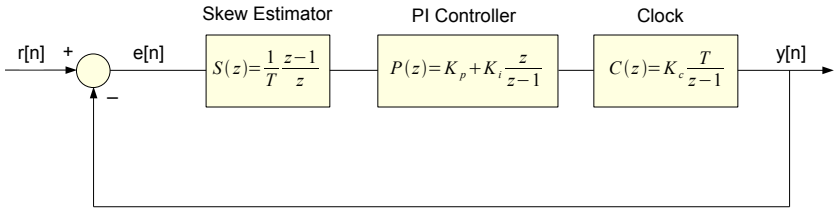


Figure 4.3: The control model for our PTP clock servo.

The closed loop system function for this controller is

$$H_{cl}(z) = \frac{K_c\left\{(K_p + K_i)z - K_p\right\}}{z^2 + \left\{K_c(K_p + K_i) - 1\right\}z - K_cK_p} \tag{4.12}$$

This function is independent of the synchronization period T, and demonstrates its robustness against synchronization message losses.

4.2.3 Time Calculation

Since we only correct the clock rate in order to drive the skew, rather than the offset, to zero, the clock may register a fluctuating offset around a constant value.

This offset, or phase error, may only be updated at synchronization (sync) instants. Its value at sync instant n is given by

$$E_n = \Delta_0 + \sum_{i=1}^{n} (\Delta_i - \Delta_{i-1}) \tag{4.13}$$

where Δ_0 is the offset at reference time $n = 0$ and Δ_i is the offset at sync instant i. The sum term accounts for accumulation in phase error caused by the imperfect rate synchronization. A leader clock must embed its current time error in the next Sync message. Consequently, the absolute global time on a follower clock at time t, $S^f(t)$, is given by

$$S^f(t) = S_c^f(t) - \left\{ E_t^l + E_t^f \right\} \tag{4.14}$$

where $S_c^f(t)$ is the reading of the follower's local clock, E_t^l is the phase error of the leader clock, and E_t^f is the phase error of the follower clock at time t.

4.3 Implementation

The IEEE 1588 protocol was implemented on top of the MAC protocol discussed in Chapter 3. Figure 4.4 shows the fixed-size MAC layer frame structure that was used to transfer data and synchronization messages among nodes. IEEE 1588 messages were uniquely identified with a value of PTP_FRAME in the frame type field.

Because the Mica2Dot did not have a capacity to transmit a Sync message with its own timestamp, the two-message model of the IEEE 1588 was adopted. No spanning tree or self-organizing algorithm was implemented. Multi-hop synchronization was therefore performed with a statically organized network.

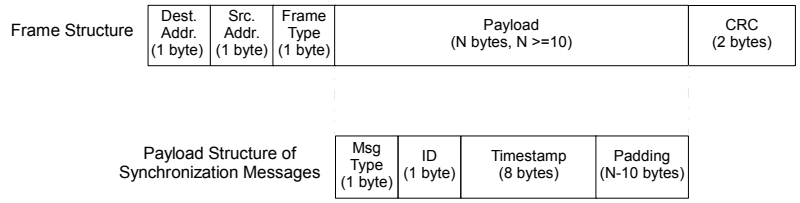


Figure 4.4: Fixed-size frame structure used to transfer data and synchronization messages. The message ID in the synchronization message payload structure was necessary to avoid associating a Follow_Up message with the wrong Sync message.

4.3.1 Protocol State Machine

Figure 4.5 is the state machine that was used to implement the synchronization on the Mica2Dot motes. The transmission time of the first Sync message by leader clocks were randomly staggered, based on the node's address, to avoid collisions and hence degraded performance. In the implementation, a boundary clock was referred to as a *leader clock*.

4.3.2 Timestamping

Timestamping is a critical part of the implementation and the method used limits the achievable precision. Two reference points are used in taking timestamps [3]: *clock timestamp point* and *message timestamp point*. Figure 4.6 illustrates these two reference points. The message timestamp point is the location in the message structure that is used as a reference point for taking timestamps. The clock timestamp point is the location in the node where timestamps are taken. A timestamp is taken when the message timestamp point passes the clock timestamp point.

The first bit of the SOF was used as the message timestamp point. The clock timestamp point for outgoing messages was in the CC1000 driver, just before the SOF byte was transmitted. The clock timestamp point for incoming messages was also in the CC1000 driver, on reception of the first non-preamble byte. The timestamp was adjusted according to the position of the most-significant-bit of the SOF byte in the non-preamble byte.

A clock resolution of 1ms was used. To obtain timestamps at the highest resolution, the counter value at the occurence of the timestamp event was used to calculate a more precise time of the event.

4.3.3 Clock Correction

The system software clock was disciplined by modulating the clock rate with the output of the PI controller. The tick rate of the software clock, represented by the parameter *nanoSecsPerTick*, is the number of nanoseconds that is added to the system time any time a clock interrupt occurs.

Large skews were observed between the motes. Hence, to speed up convergence of the clock servo, the skew of the follower clock was estimated at startup and used to correct the clock rate. As shown in Chapter 5, this resulted in eight-fold reduction in convergence time.

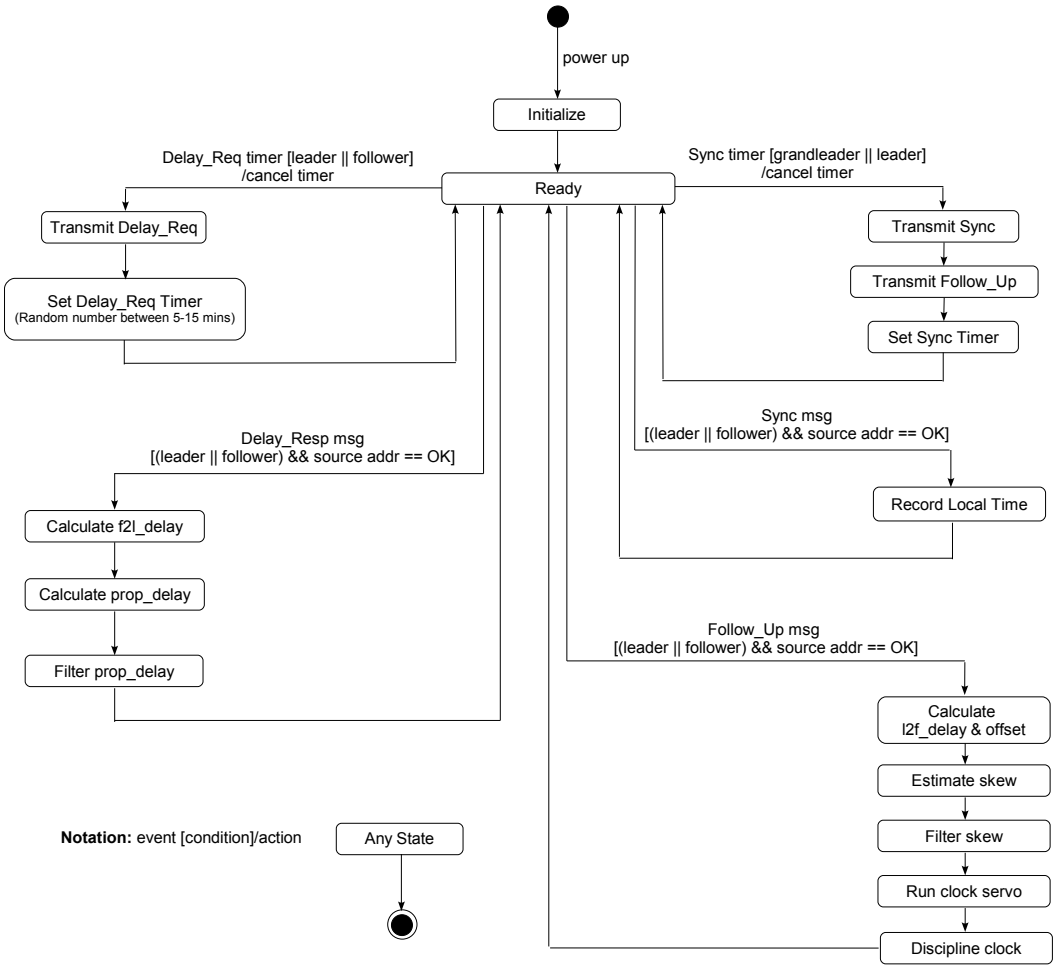


Figure 4.5: PTP synchronization protocol state machine.

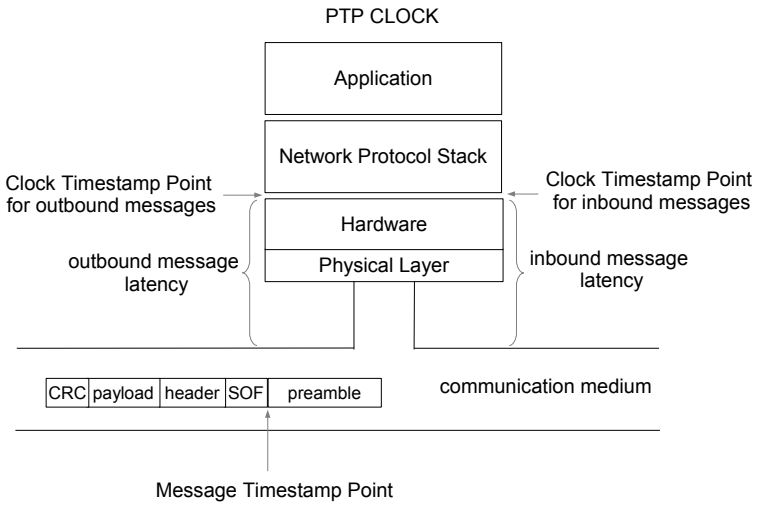


Figure 4.6: Clock and Message Timestamp Points. Both inbound and outbound messages are timestamped in the CC1000 driver.

CHAPTER V

RESULTS AND DISCUSSION

This chapter presents experimental results for the IEEE 1588 style synchronization on Mica2Dot motes. After presenting data to show that clock skew is Gaussian-distributed and therefore the Kalman filter is optimal in the mean-square error sense, the synchronization accuracy and clock stability achieved by the approach in this thesis and the one by Branicky et al. [4] are compared. Finally, the results are discussed.

5.1 Distribution of Skew and Message Latency

Two experiments were conducted to obtain the distribution of relative skew between a pair of motes. First, two motes were programmed to service the external interrupt and the counter overflow interrupt. An external signal (once every second) was then applied to interrupt pin and the number of clock cycles between two successive pulses were recorded on both motes. Next, one mote was designated as the leader and the other was the follower. On every external interrupt, the leader sent a *Sync* message with its local timestamp. The follower recorded both its local timestamp and the leader's timestamp for each *Sync* message it received. The distribution of the skew from these experiments is shown in Figure 5.1. A third experiment was conducted to

obtain the distribution of message latency. In this experiment, messages were sent between a pair of nodes and the delay between a message transmission and its reception were measured with an oscilloscope. Figure 5.2 shows the distribution of message latency with and without interrupts being enabled. All the three distributions exhibit Gaussian properties.

5.2 Synchronization Results

The technique in this thesis that filters clock skew with a Kalman filter and feeds it to the PI controller shall be referred to as *skew-PI*; and the technique by Branicky et al. [4] that filters offset with a conventional digital filter and feeds it to the PI controller shall be referred to as *offset-PI*. The results in this section compare these two techniques over the wireless link. Two metrics are used in this evaluation: (a) synchronization accuracy represented by offset standard deviation, and (b) Allan deviation of the disciplined clock. To make a fair comparison, the Ziegler-Nichols method [26] was used to tune both controllers.

5.2.1 Single-hop Synchronization

Two nodes were setup with one designated as the leader clock and the other as a follower clock. A common pulse-per-second (pps) signal was applied to an interrupt pin on each node, and the nodes printed their local time when a signal occurs.

Synchronization Accuracy

The single-hop synchronization accuracy achieved using skew-PI is shown in Figure 5.3 and that achieved using offset-PI is shown in Figure 5.4. In both cases, the synchronization period was one second. Skew-PI performs more than twice as better than offset-PI and has a lower absolute offset.

Allan Deviation

Allan deviation is a standard metric used to evaluate the stability of clocks over different averaging windows. It is defined as

$$\sigma(\tau) = \sqrt{\frac{1}{2(M-1)} \sum_{i=1}^{M-1} (s_{i+1} - s_i)^2} \tag{5.1}$$

where τ is the averaging period, s_i is skew measurement number i, and M is the number of skew samples.

Figure 5.5 shows the Allan deviation achieved by skew-PI and offset-PI. Skew-PI produces a disciplined clock with better stability in all time scales.

Effect of Synchronization Period on Accuracy

The synchronization period was varied to study its effect on synchronization accuracy. Figure 5.6 shows a plot of accuracy versus period for both methods. We could only get data upto 2 seconds for offset-PI because the servo became unstable after 2 seconds.

5.2.2 Multi-hop synchronization

Multi-hop synchronization is a more practical scenario than single-hop in engineered systems. The performance of the two methods were evaluated in a multi-hop setup. For an $n-$hop synchronization, n nodes were setup in a logically linear topology eventhough all the nodes were physically in the same wireless broadcast domain. The first node was made the grandleader, the middle nodes served as leaders, and the last node was configured as a follower. The first and last nodes were connected to an external pps signal and programmed to print their local time on the occurence of a signal. Figure 5.7 shows that while the accuracy degrades with the number of hops in both methods, offset-PI degrades much faster.

5.2.3 Convergence Time

We observed large relative skew between the motes. To speed up convergence of our servo, we used the first few Sync messages to estimate and correct the skew of the follower, before starting the servo action. Figure 5.8 shows the benefit of this correction. The clock servo converged within about 100 seconds when the skew was estimated and corrected; in contrast, it required 800 seconds to converge without the correction.

5.3 Discussion

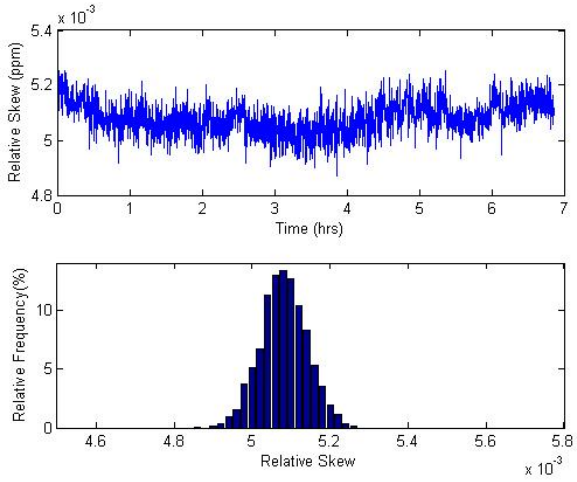
The Gaussian distribution of clock skew is important for optimal tracking by the Kalman filter. Figure 5.2 clearly shows why software only implementations expe-

rience offset spikes from interrupts. The interrupts have spread the message delay distribution. This effectively limits the accuracy that can be achieved using the offset-PI approach. However, the skew-PI approach is resistant to these spikes since they are attenuated in the conversion from offset to skew as explained in Section 4.2.2.

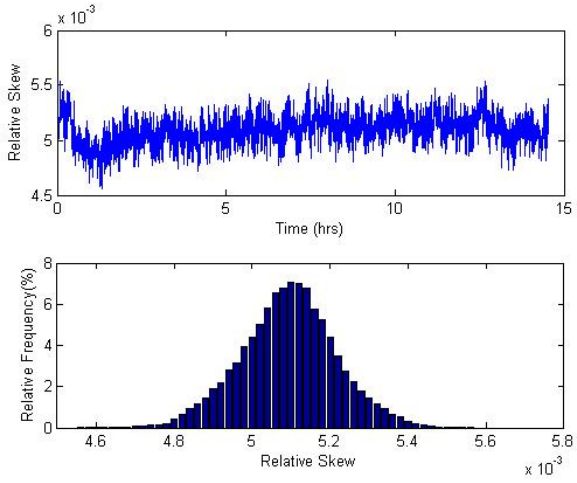
The higher accuracy obtained by the skew-PI may partly be attributed to its resistance to offset noise spikes caused by message-losses. A noise spike produced in such a case does not translate to a spike in skew. Consequently, the servo remains stable under such conditions as demonstrated by the Allan deviation plot. The same reason may account for its superior performance in the multi-hop scenario.

Viewed from a different perspective, a message-loss is equivalent to doubling the synchronization period. The closed-loop system stability of the skew-PI servo clearly shows its independence from the synchronization period. It remains stable under all practical values of the synchronization period. This is important in order to reliably trade-off accuracy for overhead, as is often the need, and still expect the servo to remain stable. This tradeoff is not possible with the offset-PI approach as its stability cannot be guaranteed for all synchronization periods without retuning the PI controller.

Fast convergence time of the clock servo is important in many applications where nodes dynamically join the network. It is therefore significant that the convergence time reduced eight-fold with the initial skew estimation and correction scheme.



(a) Distribution of relative skew using oscillator counts



(b) Distribution of relative skew from software

Figure 5.1: Relative skew measured with two methods show a Gaussian-distribution. This is important for optimal tracking of the skew by the Kalman filter.

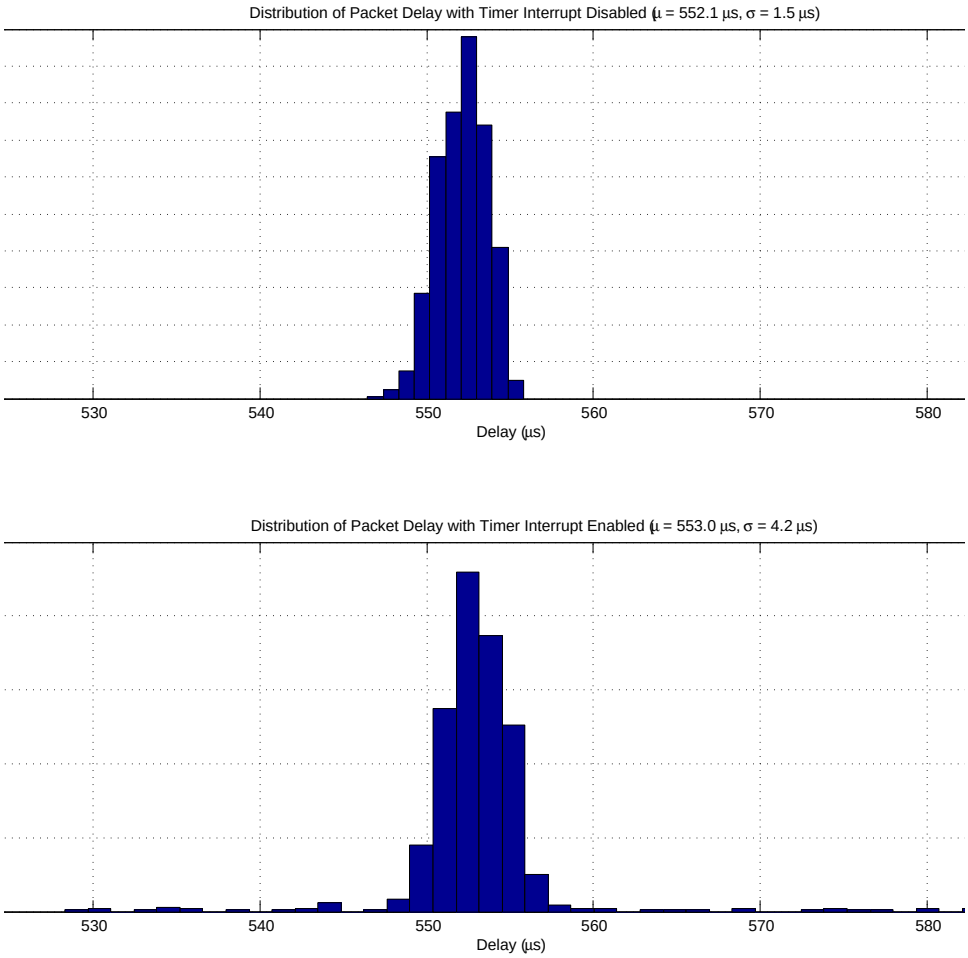


Figure 5.2: Message latency with interrupts (bottom) and without interrupts (top). Interrupts spread the distribution of the message latency: standard deviation of message latency increases from 1.5μs without interrupts to 4.2μs with interrupts.

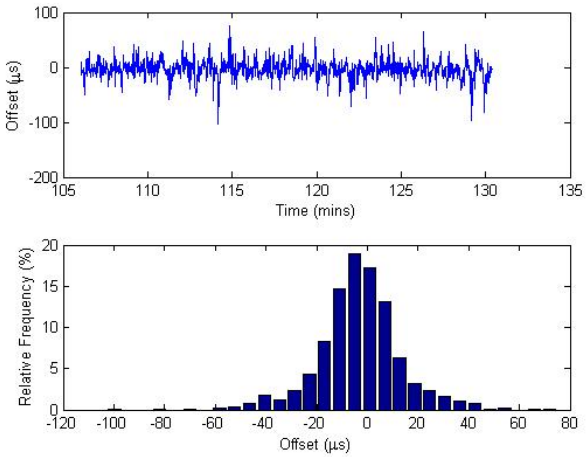


Figure 5.3: Single-hop synchronization accuracy with skew-PI method. Standard deviation is $16.8\mu s$.

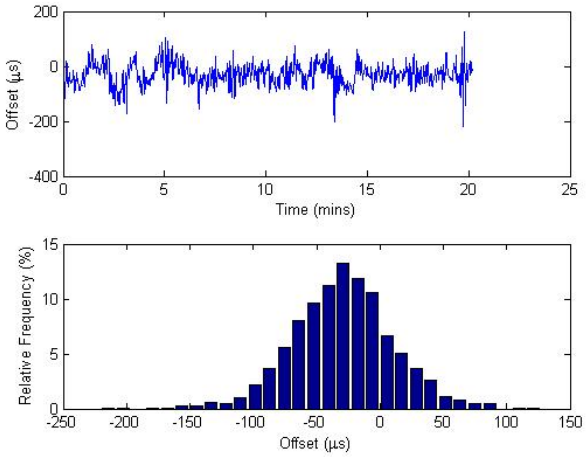


Figure 5.4: Single-hop synchronization accuracy with offset-PI method. Standard deviation is $40.4\mu s$.

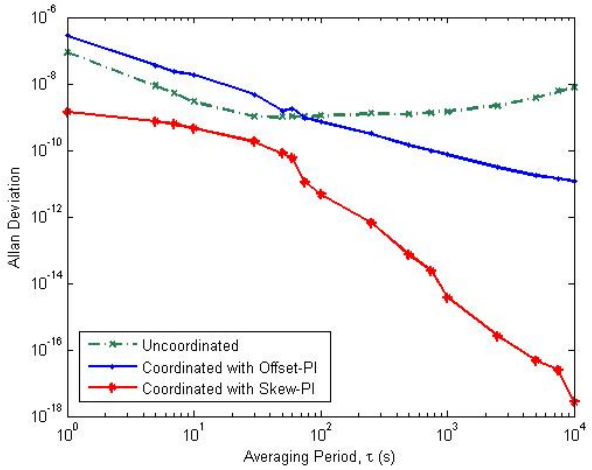


Figure 5.5: Allan deviation plots. While the offset-PI method has better stability than the undisciplined clock in the large time scale, the PTP clock servo using the skew-PI method consistently provides a far more stable clock in the wireless environment.

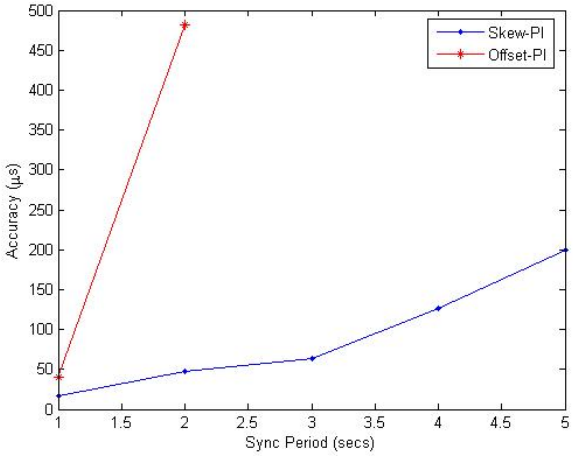


Figure 5.6: Single-hop synchronization accuracy versus synchronization period. While Skew-PI degrades slowly, offset-PI's servo becomes unstable for periods greater than 2 seconds.

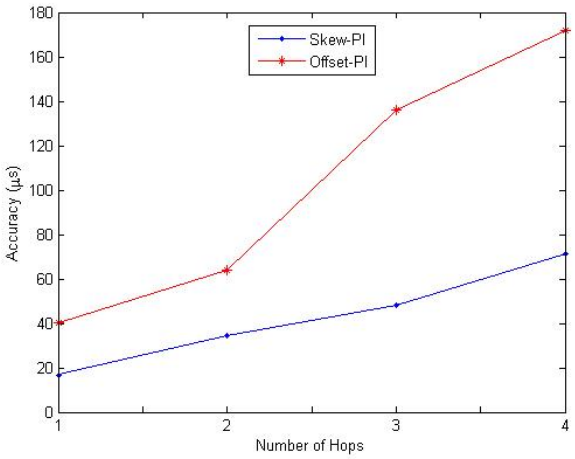
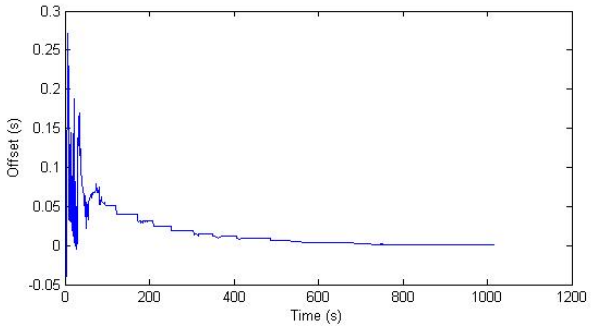
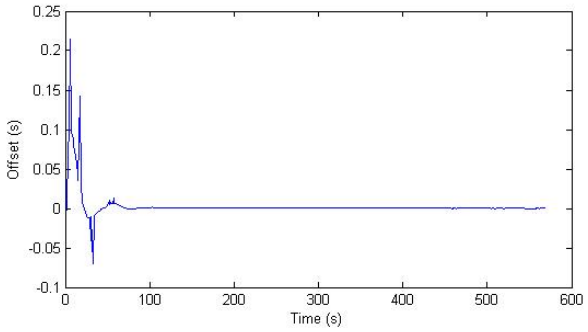


Figure 5.7: Multi-hop synchronization accuracy: Skew-PI has higher accuracy which degrades at a much slower rate with number of hops.



(a) Convergence time without initial skew correction



(b) Convergence time with initial skew correction

Figure 5.8: Convergence time with initial skew correction is about 100 seconds but it is about 800 seconds without initial skew correction. Synchronization period was one second.

CHAPTER VI

CONCLUSION

A technique for implementing IEEE 1588 style synchronization in networked embedded systems that rely on wireless media was presented in this thesis. To compensate for the poor stability of local clocks and the lossy wireless medium, a clock servo was designed that used the follower clock's skew, filtered with a Kalman filter, as the tracking signal for a PI controller. The results demonstrated that the PTP clock servo that uses the skew with a Kalman filter consistently provides a more accurate and stable disciplined clock than a clock servo that uses offset with a conventional digital filter in the wireless environment. Even with the poor stability of the oscillators used by the Mica2Dot motes, synchronization accuracies of about 17 μs standard deviation was achieved in a software-only implementation. In the future, this technique can be extended to improve fault tolerance in wireless systems and to achieve determinism in networked embedded systems. Future work may also focus on using the Kalman filter as a fusion tool for the offset from other nodes in order to improve synchronization accuracy and achieve internal synchronization.

BIBLIOGRAPHY

[1] H. Kopetz. Why do we need a sparse global time-base in dependable real-time systems? In *IEEE Symposium on Precision Clock Synchronization (ISPCS) for Measurement, Control and Communication*, Vienna, Austria, October 2007.

[2] E. Lee and S. Matic. On determinism in event-triggered distributed systems with time synchronization. In *IEEE Symposium on Precision Clock Synchronization (ISPCS) for Measurement, Control and Communication*, Vienna, Austria, October 2007.

[3] J. C. Eidson. *Measurement, control and communication using IEEE 1588*. Springer, 2006.

[4] K. Correl, N. Barendt, and M. Branicky. Design considerations for software only implementations of the IEEE 1588 precision time protocol. In *Proceedings of the Conference on IEEE-1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, Winterthur, Switzerland, October 2005. NIST and IEEE.

[5] G. Welch and G. Bishop. An introduction to the Kalman filter. Technical Report Tech. Rep. 95-041, Department of Computer Science, University of North Carolina, Chapel Hill, NC, July 2006.

[6] K. Römer, P. Blum, and L. Meier. Time synchronization and calibration in wireless sensor networks. In *Handbook of Sensor Networks*, pages 199–237. John Wiley & Sons, 2005.

[7] D. P. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, 2nd edition, 1992.

[8] et. al. M. Ali. Medium access control issues in sensor networks. *ACM Sigcomm Computer Communication Review*, 36:33–36, 2006.

[9] I. Demirkol, C. Ersoy, and F. Alagoz. MAC protocols for wireless sensor networks: a survey. *IEEE Communications Magazine*, 44(4):115–121, 2006.

[10] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the Second International Conference on Embedded Networked Sensor Systems*, pages 95–107, Baltimore, Maryland, USA, November 2004.

[11] W. Ye, J. Heidemann, and D. Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 12(3):493–506, June 2004.

[12] T. van Dam and K. Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *First International Conference on Embedded Networked Sensor Systems*, pages 171–180, Los Angeles, CA, 2003.

[13] J. Elson and K. Römer. Wireless sensor networks: A new regime for time synchronization. *ACM SIGCOMM Computer Communications Review*, 33(1):149–154, January 2003. Springer Inc.

[14] B. Sundararaman, U. Buy, and A. D. Kshemkalyani. Clock synchronization for wireless sensor networks: a survey. *Ad Hoc Networks*, 3:281–323, 2005.

[15] J. Elson. *Time synchronization in wireless sensor networks*. PhD thesis, University of California, Los Angeles, CA, USA, 2003.

[16] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi. The flooding time synchronization protocol. In *SenSys '04*, Baltimore, MA, USA, November 2004.

[17] H. Dai and R. Han. Tsync: A lightweight bidirectional time synchronization service for wireless sensor networks. *ACM Mobile Computing and Communications Review*, 8(1):125–139, January 2004.

[18] G. Gaderer P. Loschmidt and T. Sauter. Clock synchronization for wireless positioning of COTS mobile nodes. In *IEEE Symposium on Precision Clock Synchronization (ISPCS) for Measurement, Control and Communication*, Vienna, Austria, October 2007.

[19] T. Cooklev, J. C. Eidson, and A. Pakdaman. An implementation of IEEE 1588 over IEEE 802.11b for synchronization of wireless local area network nodes. *IEEE Transactions on Instrumentation and Measurement*, 56:1632–1639, 2007.

[20] MPR-MIB (Mica Platforms) User Manual. `http://www.xbow.com/support/Support_pdf_files/MPR-MIB_Series_Users_Manual.pdf`.

[21] I. Rhee, A. Warrier, M. Aia, and J. Min. ZMAC:a hybrid MAC for wireless sensor networks. In *SenSys '05*, San Diego, CA, USA, November 2005.

[22] Chipcon CC1000 Radio Datasheet. `http://focus.ti.com/lit/ds/symlink/cc1000.pdf`.

[23] T. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall PTR, 2nd edition, 2002.

[24] The Institute of Electrical and Electronics Engineers, Inc. *IEEE standard for a precision clock synchronization protocol for networked measurement and control systems, IEEE Std 1588-2002*. New York, NY, 2002. ISBN 0-73813369-8.

[25] J. C. Eidson and K. Lee. Sharing a common sense of time. *IEEE Instrumentation and Measurement Magazine*, 6:26–32, March 2003.

[26] A. O'Dwyer. *Handbook of PI and PID controller tuning rules*. London: Imperial College Press, 2nd edition, 2006.

APPENDIX

CLOSED LOOP SYSTEM FUNCTION OF SERVOS

A.1 Closed-loop System Function for Skew-PI Servo

The closed-loop servo block diagram was shown in Figure 4.3. The Open-loop system function is given by

$$H_s(z) = \frac{1}{T}\frac{z-1}{z}\left\{K_p + K_i\frac{z}{z-1}\right\}K_c\frac{T}{z-1}$$

$$= \frac{K_c\left\{(K_p+K_i)z - K_p\right\}}{z(z-1)}$$

Hence the closed-loop system function is

$$H_{scl}(z) = \frac{H_s(z)}{1+H_s(z)}$$

$$= \frac{K_cK_p(z-1) + K_cK_iz}{z(z-1) + K_cK_p(z-1) + K_cK_iz}$$

$$= \frac{K_c(K_p+K_i)z - K_cK_p}{z^2 + (K_cK_p + K_cK_i - 1)z - K_cK_p}$$

The poles are:

$$z = \frac{(1 - K_c K_p - K_c K_i) \pm \sqrt{(K_c K_p + K_c K_i - 1)^2 + 4 K_c K_p}}{2}$$

Using a normalized value of $K_c = 1$, the poles may be written as

$$z = \frac{(1 - K_p - K_i) \pm \sqrt{(K_p + K_i - 1)^2 + 4 K_p}}{2}$$

For stability,

$$\left| \frac{(1 - K_p - K_i) \pm \sqrt{(K_p + K_i - 1)^2 + 4 K_p}}{2} \right| < 1$$

Note that the stability criterion is independent of synchronization period T.

A.2 Closed-loop System Function for Offset-PI Servo

The closed-loop servo block diagram is shown in Figure A.1. The closed-loop system

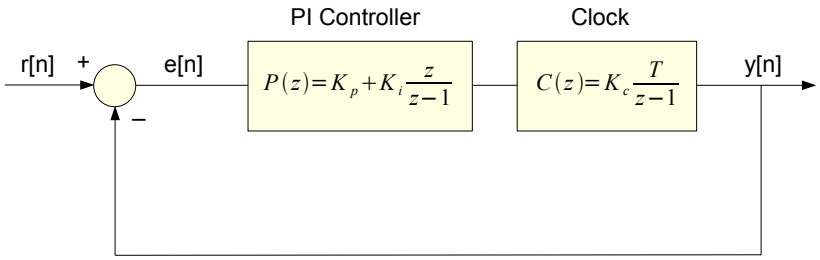


Figure A.1: The control model for offset-PI clock servo.

function presented in [3] is as follows. Its open-loop system function is given by

$$H_o(z) = \left\{ K_p + K_i \frac{z}{z-1} \right\} \frac{K_c T}{z-1}$$

$$= \frac{K_c T \left\{ (K_p + K_i) z - K_p \right\}}{(z-1)^2}$$

Hence the closed-loop system function is

$$H_{ocl}(z) = \frac{H_o(z)}{1 + H_o(z)}$$

$$= \frac{K_c T \left\{ (K_p + K_i) z - K_p \right\}}{(z-1)^2 + K_c T \left\{ (K_p + K_i) z - K_p \right\}}$$

$$= \frac{K_c T \left\{ (K_p + K_i) z - K_p \right\}}{z^2 + \left\{ K_c T (K_p + K_i) - 2 \right\} z + \left\{ 1 - K_c K_p T \right\}}$$

The characteristic equation of this system is

$$z^2 + \{K_c T(K_p + K_i) - 2\} z + \{1 - K_c K_p T\} = 0$$

and therefore its poles are given by

$$z = \frac{-(-2 + P + I) \pm \sqrt{(-2 + P + I)^2 - 4(1 - P)}}{2}$$

where $P = K_c K_p T$ and $I = K_c K_i T$. For stability,

$$\left| \frac{-(-2 + P + I) \pm \sqrt{(-2 + P + I)^2 - 4(1 - P)}}{2} \right| < 1$$

The stability of this system depends on the synchronization period T.