ABSTRACT

Title of Thesis:	THE SMART STONE NETWORK: DESIGN AND PROTOCOLS
	Amrit Bandyopadhyay Master of Science, 2006
Thesis directed by:	Professor Neil Goldsman Department of Electrical and Computer Engineering

The Smart Stone Protocol (SSP) has been developed to achieve rapid synchronization in a wireless sensor network, establish Time Division Multiple Access (TDMA) communication slots, and perform distributed sensing with global shared awareness. The SSP achieves a synchronization precision of 50μs among receivers. The sender is synchronized to the receivers using a novel scheme to identify the closest comparable times on the sender and receiver. The protocol is tightly related to events that occur in the mote hardware, and is designed to operate on resource constrained wireless sensor motes. Robust TDMA communication slots are set up based on the achieved synchronization, and an innovative algorithm is employed to maintain synchronization without sending any additional synchronization bytes. To test and validate the protocol, Smart Stones have been custom designed using commercial off-the-shelf (COTS) components, and the SSP has been successfully demonstrated on the Smart Stone Network performing an acoustic sensing application.

THE SMART STONE PROTOCOL (SSP): DESIGN AND PROTOCOLS

by

Amrit Bandyopadhyay

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2006

Advisory Committee:

Professor Neil Goldsman, Chair
Professor Shuvra Bhattacharyya
Professor Martin C. Peckerar

DEDICATION

To my parents

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor Dr. Goldsman for being a mentor to me over the last three years at the University of Maryland. I thank him for his guidance at each step of the work presented in this thesis. In addition, I would like to specially thank him for being a friend and for supporting me during a very difficult time in my life. I would like to thank Dr. Bhattacharyya and Dr. Peckerar for helping me with concepts related to circuit design, and for taking time out of their very busy schedules to peruse and provide input on my thesis. For the work in this thesis, I would like to thank Ben Funk for active participation in the project, both for his assistance in implementing the network, and for creative input in developing the protocols. I am forever grateful to my parents for their faith in my endeavors, and for always being there for me even while I am away from home. I would like to extend my thanks to all Dr. Goldsman's students Siddharth, Akin, Zeynep, Bo, Datta, Latise, Gary, Blake and Chris for being fun to spend time with, and for intense discussions on various aspects of Electrical Engineering. Lastly, I am indebted to all my friends for making the last three years a wonderful experience both academically and personally.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

1. Sensor Networks

The concept of distributed entities sharing information among each other to study and learn from the environment is an integral part of the world. For example, ants, one of the most industrious species, use their antennae to sense chemical information from their environment and from each other. The antennae not only contain information regarding the surroundings, but also convey information pertaining to which ant colony the ant belongs to. Humans have similar networks set up among themselves. The five senses may be considered to be the sensors on each person, and speech and actions may be perceived as the means to share these findings. We also possess the lucrative capability of listening to transmissions (speech) that are of interest to us, and tuning out when we hear something that is not meant for us or simply boring. These analogies though rather digressive at a first glance have the potential to present solutions to the challenges faced while building small electronics that independently monitor the environment as part of a distributed network.

This chapter contains a comprehensive discussion on Wireless Sensor Networks (WSNs). These networks have been projected to be the solution in several applications which require a large area to be continuously monitored. This is achieved by distributing large numbers of low cost wireless sensor nodes in the area to be monitored. These nodes continuously sense the environment, communicate events to each other, and route their information to a remote base station. The most significant challenge in accomplishing this goal is the power constraint on these small, low cost nodes. This renders protocols to achieve efficient low power communication among these wireless sensor nodes an open research topic. The communication protocols gain even more importance from the fact that the greatest amount of power is consumed by the transmission and reception tasks performed by these nodes. Hence, protocols that schedule and distribute these tasks among the network nodes while minimizing power consumption are necessary to justify the use of wireless sensor networks to realize the application. The limitations of WSNs to be considered while developing communication and information sharing protocols are stressed in this chapter, and an objective analysis of existing clock synchronization and medium access protocols is performed. The analysis served as both motivation and guidance for the development of the Smart Stone Protocol (SSP).

The SSP is an innovative Time Division Multiple Access (TDMA) based communication protocol. The SSP also achieves rapid synchronization among sender and receivers, establishes TDMA communication slots, and also incorporates sensing and data sharing tasks. Finally, the Smart Stone Network, a network of inch sized wireless sensor nodes, has been developed, and the SSP has been successfully demonstrated on the network implementing an acoustic sensing application. The Smart Stones serve as a platform for testing and validating the algorithms developed. However, more importantly, the hardware features and restrictions of these motes played a vital role in the development of the protocol.

1.1 Wireless Sensor Networks (WSNs)

Real-world phenomena surround us all. Monitoring these phenomena that present themselves as signals, provides us with an awareness of our surroundings, a means to automate tasks that cannot be efficiently performed by people, and most importantly, can generate warnings when people are faced with natural or devised threats. The last few years have resulted in tremendous advances in micro-electromechanical (MEMS) technology [1] leading to the development of low power, and low cost tiny sensors. These novel sensors coupled with low power transceivers and microcontrollers have realized the concept of the Wireless Sensor Node. Wireless Sensor Networks (WSNs) are large scaled networks of these wireless sensor nodes aimed at performing a specific application. In general, most WSNs comprise several low cost, tiny motes, equipped with one or more sensors to monitor the environment, a microcontroller to gather and process the information, and a transceiver to communicate these readings to other nodes in the network. Transceivers in wireless sensor nodes are usually only capable of receiving or transmitting at a time. The mechanism of gaining access to the medium to transmit information is decided by the *Medium Access Control* (*MAC*) *Protocol* used, and messages are passed from sources to destinations (*sinks*) along paths determined by the *Routing Protocol*. Finally, the messages after data fusion and aggregation at nodes are forwarded to a remote base station by the sinks, where the monitoring results can be viewed. WSNs are already projected to be the solution in several military [2], environmental [3], medical [4], and home monitoring [5] applications. WSNs, which might be only a couple of years from being integral parts of our lives, pose several challenges [6] to be overcome for efficient use. Most of these networks are envisioned to be used in areas where power lines cannot be provided. Irrespective of the application, the algorithms developed to govern the functioning of these networks must take into account the inherent limitations of energy, bandwidth, size, and cost present in WSNs.

1.2 Limitations in Wireless Sensor Networks

Though faced with the daunting task of collecting and processing intense amounts of data, while communicating this data to the entire network, Wireless Sensor Nodes must achieve this by optimally using the limited resources they are equipped with. The most significant of these limitations are energy, bandwidth, and hardware. These limitations render traditional networking protocols used in wired systems, and wireless systems that are line-powered, impractical for direct application in WSNs. This necessitates the development of algorithms specific to WSNs acknowledging their restrictions and specific applications. In particular, all these limitations are correlated to each other, and proposed algorithms must consider the entire set of inherent limitations. This section summarizes the challenges faced by these networks.

1.2.1 Low Power

The most pressing challenge of all is realizing the application in spite of the limited energy possessed by the network. Majority of the applications require the nodes to be distributed in areas where it is impossible to wire these nodes to a power source. In addition, they are intended to operate without human intervention, introducing the need of a battery source of power which should not rely on being replaced when the battery life terminates. Applications [2-5], however, require the node to function for years on the allotted battery source. This imposes the need for ultra low power operation. The energy in the network may be viewed as whole, or for each node separately. Each node has some initial energy reserve and may perform tasks of sensing, signal processing, communication of its own data, and routing of other nodes' data through the network. Draining the power of nodes in a certain area will degrade the routing efficiency of the network, increase the energy consumption of the remaining nodes, and result in a cascading decay of the network. Also, as nodes "die", certain areas are unmonitored. Assigning high power tasks to a few special nodes is beneficial if nodes with higher power resources exist, and can be strategically placed in the network, which is usually not the case in WSNs. To employ these techniques, schemes of rotating the roles of "special nodes" must be proposed to ensure a uniform loss of power through the network. The network lifetime does not depend on a few nodes having power reserves, but on the entire network performing its intended function over the application region during its lifetime.

The power limitation discussed above also restricts the bandwidth. Experiments [7] have shown that the greatest sources of power consumption are wireless transmission and reception. This poses strict limitations on transmission length, the length of the report sent by each node, and the transmission bandwidth. In addition, it introduces the need for multi-hop algorithms [8], since in several applications a single transmission to all nodes in the network can often be more power consuming than several small transmissions, cascading the information to each sink in the network.

1.2.2 Low Bandwidth and Small Transmission Length

Given that the amount of power required to transmit data is much larger than that required to process data, a direct constraint is imposed on the bandwidth that can be utilized for each transmission. Presently, wireless communication in WSNs is restricted to a data rate in the order of 10–100 Kbits/second [9]. Pottie and Kaiser [7] have argued that the energy required to transmit 1Kb over 100 meters, 3 joules, can be used by a 100 MIPS/W general purpose processor to execute 3 million instructions. This highly cited result, though specific to the authors' assumptions [7], is a strong indication of the dominating factor in power considerations. Bandwidth limitation directly affects message exchanges among sensors. In order to efficiently achieve data sharing and synchronization of nodes, practical algorithms proposed must achieve the same using small packets and by avoiding several synchronization messages.

The power and bandwidth limitations require each node's transmission report to be designed intelligently since each additional byte requires a large amount of energy to be conveyed to every destination node. In addition, designers must remember that in a multi-hop scenario each node might be transmitting the reports of several nodes, hence multiplying the power required to achieve this. In all cases except direct communication with a base station, the communication power is a combination of transmit and receive powers for each hop. In cases where synchronization is required, the number of bytes required to synchronize clocks must be minimized and preferably attached to the sensor report rather than require each node to transmit separate synchronization messages.

1.2.3 Limited hardware (low cost and small size)

Some WSNs are projected to comprise several thousands of tiny nodes which are distributed to cover and monitor the area of concern. Their low cost facilitates the existence of large-scale networks, and renders a node expendable if it fails. The small size and low cost however, limit the amount of electronics that can be accommodated on these tiny boards within the node budget, and be run by the limited energy. The computations must hence be performed with fewest instructions possible, and with limited memory. The limitations also allow transceivers with short transmission range (usually less than 50 meters) necessitating the use of multi-hop transmission for global awareness. Also, network protocols such as Frequency Division Multiple Access (FDMA) where each node transmits on a separate channel, requiring transceivers which can receive a very large number of frequencies, seem rather impractical.

1.2.4 Limited network connectivity and dynamic connectivity

Wireless mediums are subject to interference from external sources which may corrupt messages from the transmitting node. The limited range makes global sharing of information at low power a huge challenge and often requires multi-hop routing. An increasingly popular application of WSNs is mobile distributed computing, where nodes are continuously moving. In such networks, nodes dynamically change the nodes to which they can reliably transmit to, requiring the network nodes to dynamically determine routing paths. This limitation introduces the need for a highly adaptive MAC protocol, robust to new nodes entering and leaving the transmission range of each mobile node. In addition, given the high probability of message corruption, the protocol should ensure that network nodes do not interfere with each other, since that would add to the number of re-transmissions required for external interference.

1.2.5 Large-scale network and remote monitoring

Almost all WSN applications draw their strength from the size of the network by scattering these nodes to cover a large application area, and densely populating certain zones. For these nodes to communicate with each other, the nodes must dynamically determine source to sink routing to share information. Simple peer-to-peer broadcast solutions though important, usually form only the method of communication within clusters. More complex protocols are required to extend the local communication to a network-wide communication method. Also, all the sensor information, and results of processing are usually viewed at some remote monitoring center, leaving the nodes unattended for periods up to years. This requires the network to find the lowest power solution to broadcast this information to the remote monitoring center. The tradeoff is to determine how many nodes should participate in the high power transmissions to cover the required distance since these nodes will drain their batteries faster.

1.3 Medium Access Control (MAC) and Routing in WSNs

As in all shared-medium networks, Medium Access Control (MAC) is an important technique that ensures the successful operation of the network. The fundamental task of the protocol is to assign transmission times to the competing nodes and prevent collision and interference. Commonly used MAC protocols are Time Division Multiple Access (TDMA), Code-Division Multiple Access (CDMA), and contention-based protocols such as Carrier Sense Multiple Access (CSMA). The MAC protocol resides in the Data Link Layer, which itself is not only responsible for fair distribution of resources, but also for providing frame detection and error control. Most of the currently pursued MAC protocols fall under the above categories, or hybrids of these protocols. A detailed discussion of MAC protocols is available in the survey by Akyildiz et al. [11]. Some prominent protocols that reveal and solve critical issues in MAC design are discussed in this section. Techniques to extend the MAC protocols using a global routing scheme are also covered while discussing the MAC protocols.

1.3.1 Self-Organizing MAC for Sensor Networks (SMACS)

The SMACS protocol [12] proposed by Sohrabi et al. is a seminal paper in the area of MAC protocols and message routing in WSNs. Their publication is more conceptual in nature rather than a detailed solution to existing problems, and is discussed in detail in this section since it addresses a majority of scenarios arising in WSNs. Recognizing the need to minimize power in large WSNs the authors dismiss the use of CSMA and propose a TDMA solution to control the communication. Using TDMA, there are two types of slot allocation schemes, *node* activation, and *link* activation. In node activation, each node is assigned a specific slot during which it can transmit to all its neighbors, whereas link activation implies the allocation of time slots based on directed links between two nodes. As shown in figure 1.1b, each node executes a transmission/reception schedule periodically every T_{frame}, which is the length of its *superframe*, and is a MAC parameter. At startup the nodes wake up at random times, and listen to the medium for a random period of time. If within this period the node receives a *TYPE1* message, a connection invitation, it responds with a *TYPE2* message to accept the invitation. If the listening time elapses without hearing an invitation, it sends its own *TYPE1* invitation message. This message may be received by one or more nodes which respond with *TYPE2* messages. If collisions do not occur, the *inviter* receives the *TYPE2* messages from all nodes in range, and must choose only one invitee to connect to. This is done on a first arrival basis, but may take other parameters such as Received Signal Strength Indication (RSSI) into account. A *TYPE3* message is sent by the inviter to inform the nodes of the chosen invitee. The invitees who are not chosen turn off their transceivers for some time and restart their search procedure. The chosen invitee responds with a *TYPE4* message, and a pair of short test messages exchanged between the two finalizes a permanent link. This transfer of messages is illustrated in figure 1.1c. A concern of the proposed algorithm is finding free slots in the existing superframe, and choosing a frequency channel to transmit on, since neighboring, concurrently scheduled links will interfere with each other. The method assumes a large number of channels, and the frequency channel and slot schedule of the link is determined within the message transfers:

TYPE1: Invitation message by inviter

TYPE2: Response to an invitation; contains information of invitee's attachment state

TYPE3: Response to *TYPE2* with *ID* of chosen invitee. If both inviter and invitee are attached to other nodes, inviter's schedule is sent too. If the inviter is attached and the invitee is not, a randomly selected proposed channel and chosen slots are communicated to the invitee.

TYPE4: Response to *TYPE3* message. If inviter is unattached, the invitee chooses and communicates both the chosen frequency and the slots. If both are attached, the invitee uses information of both nodes to choose the slots and the channel.

These messages ensure the reservation of two consecutive, overlapping time slots in each node's superframe, one for transmitting, the other for receiving on a particular frequency channel. Further connections are made using the same scheme only if overlapping slots are found in the pair's superframes.

Sohrabi et al. also acknowledge the presence of a low density of mobile nodes within the stationary WSN. The mobile nodes must establish connectivity with the stationary nodes, and may form part of the source-sink routing path. The connectivity is established using a novel Eavesdrop and Register (EAR) algorithm, and the responsibility of connecting and disconnecting is assigned to the mobile nodes to reduce the work load on the stationary nodes. Assuming that the stationary nodes transmit infrequent invitation messages to discover new nodes, the mobile node eavesdrops on these messages, calculates Signal-to-Noise ratios (*SNRs*), maintains these in a registry, and based on thresholds decides to connect to or disconnect from a particular stationary node. If the threshold is met, the mobile node sends a Mobile Invite (*MI*) message in response to the stationary node's Broadcast Invite (*BI*). If the stationary node has communication slots available it accepts the invite with a Mobile Response (*MR*), and the two communicate until the *SNR* falls below a threshold, upon which, the mobile node disconnects from the stationary node by sending a Mobile Disconnect (*MD*) message. Failure of stationary nodes to respond to *MI's* causes the mobile node to degrade them from a *PENDING* to a *NOT-CONNECT* status for future reference.

To extend their algorithm to accommodate the entire network, Sohrabi et al. propose an adaptive local routing scheme for cooperative signal processing, employing a single winner election (SWE) algorithm and a spanning tree (ST) algorithm. Phase I involves nodes sensing a signal, collecting data, and preprocessing this data to determine whether to participate in the cooperative processing. In phase II the intention to participate is communicated to neighboring nodes. Phase III elects the Central Node (CN) which performs further data processing after receiving information from several nodes via a minimum spanning tree also formed during the same phase. Nodes after evaluating an election criterion invoke a voluntary delay, and then declare themselves as CN candidates with a first batch of *Elect* messages. The receiving nodes compare these CN candidates with themselves and respond with a second batch of *Elect* messages which carry the result of the comparison. These messages spawn further message exchanges, with better candidates being registered, and inferior ones discarded, until the election results have diffused through the entire network. The quality of a candidate's election criterion can be used by it to determine its initial delay, allowing better candidates a head start to prevent message overheads for losing candidates. The result is a minimum hop spanning tree rooted at the elected CN.

The authors do an excellent job of highlighting critical requirements in WSNs and possible approaches to solve them. The conceptual nature of the paper however necessitates several details such as thresholds, frame lengths, superframe structure, election criteria, and desired number of CNs to be worked out before possible implementation. In addition, details for accommodating the routing protocol within the proposed superframe structure are not provided. The link activation scheme involves large overhead since the nodes cannot take advantage of available bandwidth or broadcast in the medium, and must retransmit the same information to each neighbor, draining energy, and significantly increasing the size of their superframes. In addition, the assumption of having a very large number of frequency channels may not always be realistic in WSNs. The concepts proposed by the authors, however, provide an excellent understanding of WSNs, and a starting point for development of more detailed MAC and routing protocols.

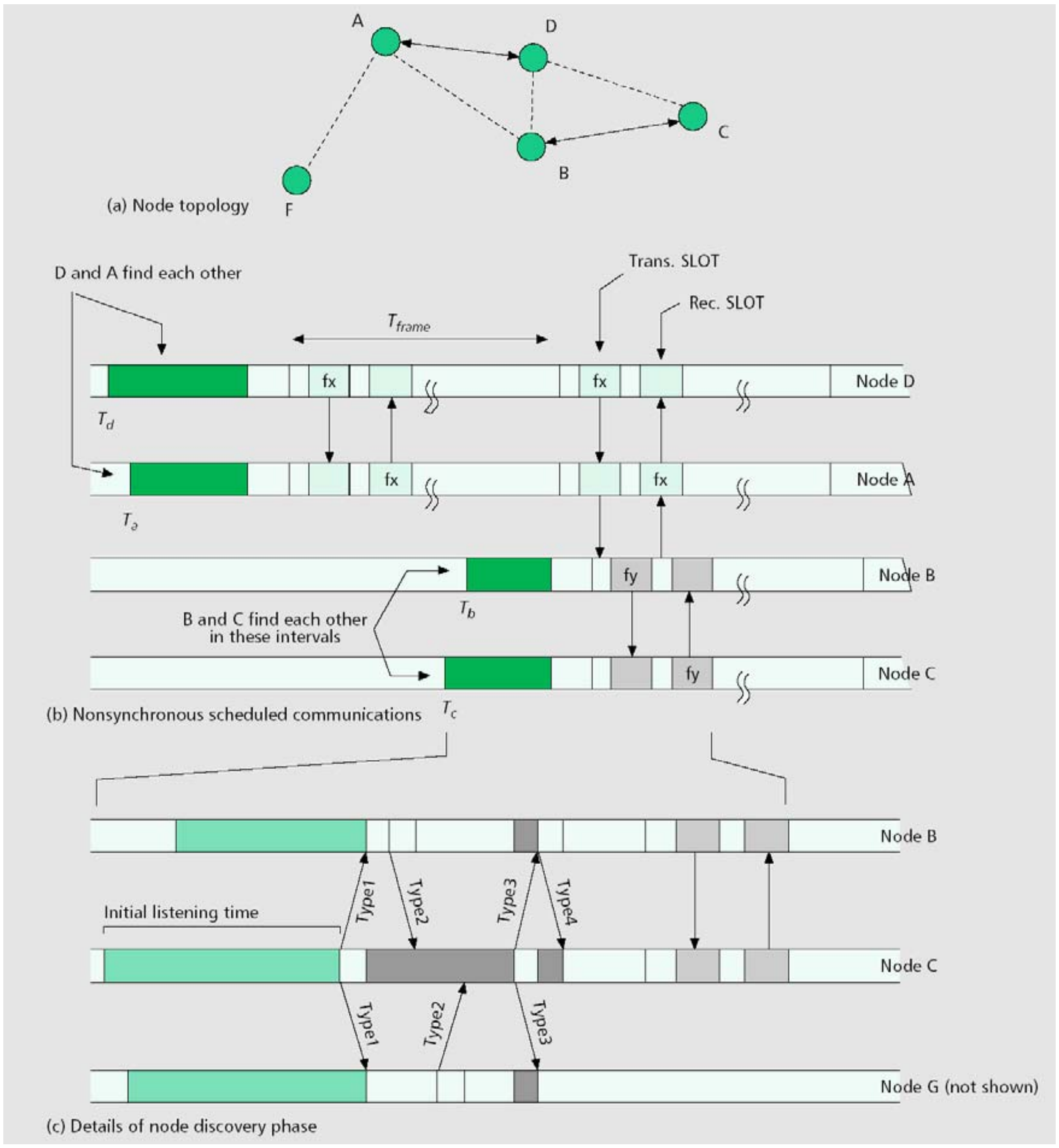

Figure 1.1 [12] a) Link discovery between nodes A-D and B-C. b) Superframes in A, B, C and D with random wake times and slots reserved for transmit and receive. c) Node discovery using 4 message *TYPES*.

1.3.2 Sensor MAC (S-MAC)

Sensor-MAC (S-MAC) [13] is a CSMA based MAC protocol directed towards multi-hop routing, and attempts to reduce energy consumption from collisions, idle-listening, overhearing, and control overhead. The concept is driven by the fact that in many sensor networks, nodes have long periods of idle time if no sensing event takes place, and idle listening may consume 50-100% of the energy required for receiving. The authors point out that Stemm and Katz measure idle : receive : send ratios of 1:1.05:1.4 [10], and Digitan Wireless Local Area Network (WLAN) module specification shows idle : receive : send ratios of 1:2:2.5 [34].

Taking advantage of this phenomenon, the S-MAC puts nodes into periodic sleep states. A frame, hence, comprises a cycle of *listen* during which nodes wait for other nodes to communicate with them, and then the sleep cycle. To reduce control overhead, S-MAC attempts to make neighboring nodes listen and sleep at the same time if this agrees with the multi-hop schedule. Before sleeping, a node waits for certain amount of time to hear the schedules of other nodes. If none are received, the node randomly chooses a sleep time, deems itself a *synchronizer*, and broadcasts its schedule in a *SYNC* message, indicating it will sleep in t seconds. If the node receives a schedule from a neighbor before choosing its sleep time, it is a *follower*, and adopts the received schedule, invokes a random delay of t_d seconds, and broadcasts this schedule, indicating its sleep time to be in t-t_d seconds. A node that receives a differing schedule after broadcasting its own, adopts both, waking up at its own, and its neighbor's *listen* times. This information is re-broadcasted before sleeping. This scheme requires synchronization which is achieved by adjusting timers after *SYNC* receptions. The synchronization requirement, though, is much less stringent than in TDMA protocols. The listening interval is divided into two periods, one to receive *SYNC* packets, the other to receive a Request to Send (RTS).

The RTS and Clear to Send (CTS) mechanism coupled with virtual and physical carrier sense is used by the S-MAC to allocate the medium. As the names suggests, RTS is an indication that a node wants to transmit to another node. The first node to send an RTS to the receiver wins the medium, and receives a CTS if the receiver is not scheduled to receive from another node. This alleviates the hidden terminal problem where two transmitting nodes with the same destination node, but out of each other's range, will tend to transmit together since they do not sense each other's RTS. They will however, hear the receiving node's CTS, and the *losing* node will wait before trying to retransmit. To perform virtual carrier sense, each node checks the *duration field* in messages destined to other nodes, sets a timer for the duration, and refrains from transmitting till the timer decrements to 0. Physical carrier sense is performed at the physical layer by continuously listening to the medium. Unicast data transmissions are achieved using the sequence of RTS/CTS/DATA/ACK where an ACK is a data message acknowledgment sent by the receiver.

Overhearing avoidance is the second technique used by the S-MAC to prevent the waste of energy in listening to long data packets intended for other nodes. To avoid overhearing, nodes go to sleep after hearing RTS or CTS packets between other nodes, and must perform this when their neighbor is either the receiver or the sender. This is necessary since the node might interfere with subsequent RTS, CTS, DATA or ACK packets, necessitating the other nodes to retransmit their information. Finally, the authors acknowledge the disadvantages of retransmitting long messages in case of collisions, and propose sending them as several shorter packets with the trade-off of increased control overhead. The messages are however sent with only one RTS/CTS pair in the beginning, and have the capability of extending their previously reserved time if fragment retransmission is required. This reduces the fairness of access in the medium and is an important feature of the protocol.

Ye et al. do an excellent job of addressing power issues in CSMA networks. Their sleep mechanism saves network wide power at the cost of message latency. Using the S-MAC a propagating message might have to wait at each hop for the intended receiver to wake up, introducing possibly large delays in a path already hindered by carrier sensing, backing off, transmission/retransmission, and processing delays. Finally, though the authors address the hidden terminal problem, in dense topologies, the possibilities of collisions still remain high in this scenario, especially when nodes wake up after a CTS or an intermediate ACK has been sent. This problem is the inherent disadvantage of all contention based schemes.

1.3.3 Low-Energy Adaptive Clustering Hierarchy (LEACH) Routing Protocol

LEACH [15] is one of the most popular hierarchical routing algorithms in sensor networks, and uses a combination of MAC protocols to resolve medium access and routing requirements. It merits thorough discussion because of the principles used to extend the protocol to the entire network and the crucial simulation results presented as motivation for the algorithm. Heinzelman et al. propose LEACH, a cluster based routing protocol, which randomly reassigns cluster heads to uniformly distribute the energy load among network nodes. To validate the need for their new algorithm, the authors have performed analyses and simulations comparing two protocols, direct communication and minimum transmission energy (MTE) routing. Their results indicate that direct communication, where nodes directly transmit to a base station, instead of each other, may be an acceptable or even optimal solution if the base station is very close to the network nodes. When the base station is far away, as is the envisioned scenario in most WSN applications, the large amount of transmit power required on each node will drain the energy reserves of each node, and result in a shortened lifetime of the network. In addition, their simulations revealed that nodes furthest away from the base station were the first to "die". The MTE routes messages from the source to the base station via multiple hops via network nodes, replacing the single transmission with several small transmissions and receptions. MTE simulations indicated that the nodes closest to the base station experience the heaviest traffic as the messages converge towards the base station and die out quickly, leaving the area unmonitored, and increasing the energy consumption in the remaining nodes. Figure 1.2a is a graph of the number of sensors alive against the elapsed time in the network for these two communication schemes. Finally, a clustering scheme was considered which uses local base stations to collect data from the cluster nodes and transmit it to the global base station. This would work if the local base stations had significantly superior power reserves, but sensor networks are not usually equipped with strategically located *supernodes*.

The LEACH algorithm, proposed to address all the above studied issues, is a clustering scheme beginning with a cluster set-up phase, followed by a steady state phase for data transfers to cluster heads, and the remote base station. The set-up phase begins with an "advertisement phase" in which each node declares itself a cluster head, say *CH*, for the particular round based on a function of desired cluster heads in the network, and the number of times the node has been a cluster head in previous rounds. The nodes generate a random number between 0 and 1. The node declares itself a cluster head for the current round *r* if this number is less than a threshold *T(n)*:

$$T(n) = \frac{P}{1 - P * \left(r \bmod \frac{1}{P} \right)} \qquad \text{if } n \in G \tag{1.1}$$

where *G* is the set of nodes that have not been cluster heads in the last *1/P* rounds, and *P* is the desired percentage of cluster heads in the network. If $n \notin G$, *T(n)* equals 0. Thus, a node can be a cluster head only once in each series of *1/P* rounds starting with round 0. All nodes are eligible to be cluster heads after the series of *1/P* rounds is over, ensuring that the cluster head duties are uniformly rotated among the sensor nodes. Each node that declares itself a cluster head, next, transmits an advertisement in a CSMA "cluster-head-advertisement" phase. Non cluster heads keep their transceivers on in this phase receiving all in-range advertisements, and choose the cluster head with the highest RSSI value to minimize energy required to transmit to the cluster head. In a second CSMA phase, the non cluster heads transmit information regarding which *CH* they are joining. Upon reception of all nodes joining the cluster, the *CH* computes a TDMA schedule based on the number of nodes joining the cluster and broadcasts this schedule to the cluster nodes. Once the cluster is set up, cluster nodes transmit their data to the *CH* at their scheduled times, while the other cluster nodes turn themselves off. The cluster head compresses the entire cluster's information into a single signal and uses a high-energy transmission to communicate this to the base station. These phases are performed for each round. The authors resolve the problem of neighboring clusters interfering with each other using different CDMA codes for each cluster. In addition, the entire protocol can be extended to form hierarchical clusters where cluster head nodes communicate with "super cluster head nodes" and so on.

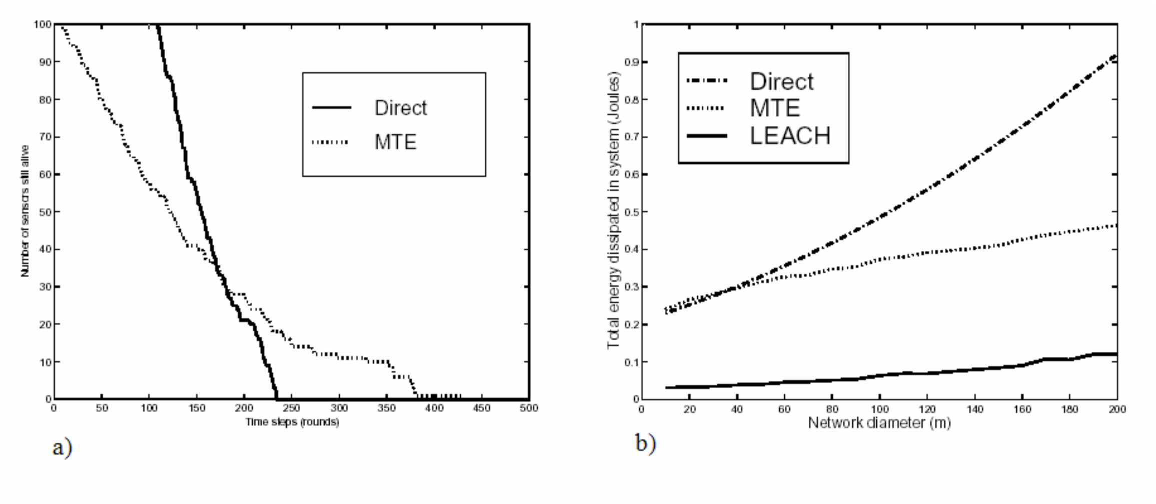

Figure 1.2 [15] a) Number of nodes alive Vs Time elapsed for direct communication and MTE. B) Energy Dissipation Vs Network diameter for Direct, MTE and LEACH

Simulations of LEACH by the authors resulted in approximately 5% cluster heads being the lowest energy solution for the simulated case, reducing communication energy up to 8x compared with direct communication and MTE. Also, the first node died over 8 times later, and the last node 3 times later than the same occurrences in the other methods. Increasing the number of cluster heads beyond the best solution reduced energy efficiency, since more than required cluster heads were performing high power transmissions. Reducing the percentage below the best solution resulted in energy loss due to higher power communication within the cluster. The LEACH protocol is effective in uniformly saving energy in the network while achieving global routing. Node sleeping is incorporated into the algorithm further minimizing energy consumption. The assumption that every node in the network can perform a high powered transmission to the base station is a limitation of the algorithm as this is not the case in several WSNs. Also, percentage of cluster heads required may change over time due to node failures, and addition of new nodes. Cluster head declaration based solely on probability and fairness may result in nodes close to each other being cluster heads in a round and reduce the effectiveness of the algorithm in these rounds. Finally, using CDMA codes to prevent inter-cluster interference increases the bandwidth requirements of the network. The protocol, however, is an excellent scheme to extend TDMA based cluster communications to the entire network, and modifications of the LEACH tailored to a specific application will result in considerably low power networking.

1.3.4 Power Aware Clustered TDMA (PACT)

The PACT protocol [14] proposed by Pei et al. addresses some of the limitations of the LEACH protocol, by rotating the roles of cluster heads and gateway nodes using energy criteria. PACT reduces energy consumption in a TDMA network, using passive clustering and turning nodes off during inactive traffic periods. Taking advantage of the fact that all nodes turn on their transceiver during the control phase of every TDMA frame, the clustering status information is piggybacked on the control messages during this phase. Each node decides its own state: cluster head, gateway node, or ordinary node based on the information received from its neighbors. This state is communicated to the neighbors in the control phase to define the clusters. Pei et al. define a fourth state named the Low Energy State (LES). Gateway nodes and cluster heads whose energy falls below a certain threshold, change their state to LES to indicate that they do not participate in the passive clustering until their batteries are recharged, if possible. They, however, continue their sensor operations. Within the control packet, each node includes the IDs of up to *n* cluster heads, and keeps a track of the cluster heads reported by its neighbors. The node which receives transmissions from the highest number of cluster heads will be the gateway node. Deciding and communicating this information within the control phase, limits the number of gateway nodes between neighboring clusters, and allows other potential gateway nodes to save energy for future use. The gateway node relays data messages between clusters.

Unlike the SMACS protocol [12], the PACT protocol uses a node activation TDMA scheme where each node is assigned a slot to broadcast its information. As shown in figure 1.3, each frame comprises control mini slots and data slots. A node broadcasts in its data slot allocation, and specifies the destination addresses for these data slots during its designated control mini slot. The slot allocation is performed giving cluster heads and gateway nodes selection priority, implying that ordinary nodes will yield their slots to accommodate the higher priority nodes. During this phase each node also determines the slots during which it is a destination, receives at these times, transmits during its own slot, and sleeps for the rest of the frame. If the control phase slot information received indicates a broadcast, all nodes stay awake when the slot occurs.

PACT simulations revealed that the protocol can improve the lifetime of the network more than 5 times compared to the IEEE 802.11 standard [27]. In addition, the network lifetime can be more than doubled with 1.5 times increase in node density. The protocol proposes an effective method of extending TDMA schedules to the entire network, and the clustered structure lends itself well to incorporating higher level routing protocols to minimize global energy. However, as pointed out in [13], TDMA requires much tighter synchronization than CSMA to realize the data slots which are already of a very short length. The biggest concern regarding the PACT protocol is the presence of even smaller synchronized control mini slots. These mini slots will require extremely precise synchronization to be achieved before the control mini slots can operate without collisions. Collisions in these mini slots will defeat the purpose of the entire protocol, because all scheduling information is contained in the control packets transmitted in these slots. In addition, the sleep cycle is strictly dependent on correct reception of these packets, meriting implementation on physical sensor network platforms to determine if the PACT protocol is feasible with hardware limitations such as transmit-to-receive switching times.

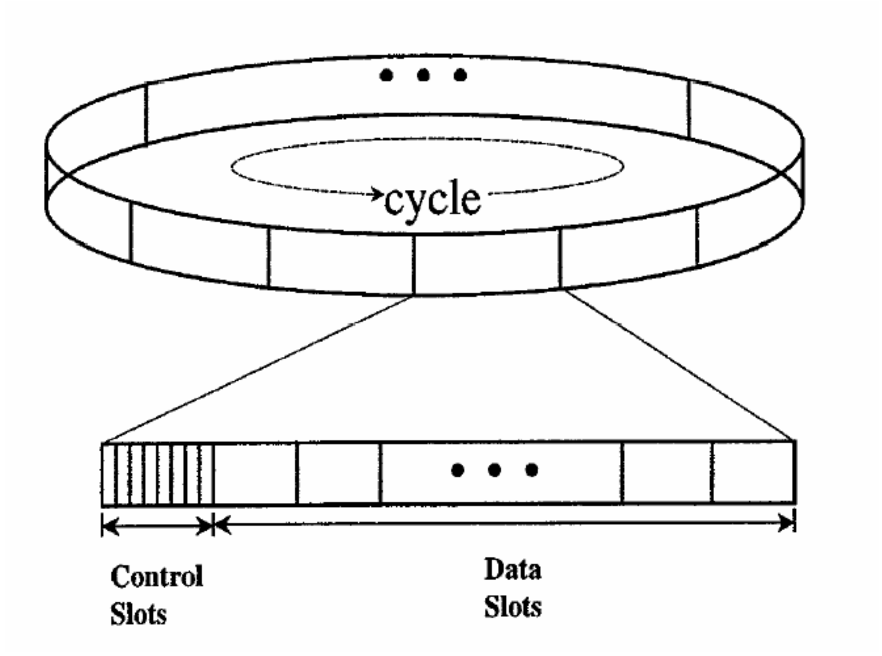

Figure 1.3 [14] PACT TDMA structure

1.4 Clock Synchronization in WSNs

The discussion in Section 1.3 on MAC and Routing protocols brings us to the important issue of clock synchronization in WSNs. One of the shortcomings of most publications proposing MAC protocols is the absence of clock synchronization details. Even in TDMA schemes, where clock synchronization is crucial, the required level of precision is simply assumed, and often not even addressed. Some form of clock synchronization or awareness of the local time on the transmitting node is critical irrespective of which MAC is chosen to govern the sensor network. All the proposed applications of wireless sensor networks involve monitoring sensor readings and communicating events to the other network nodes, or sinks. The deductions made during data fusion are sensitive to the time at which an event occurred at each node. This makes the need for clock synchronization extremely crucial to the network. In addition, CSMA though very promising, and maybe easier to implement than TDMA, does not eliminate the possibilities of collision at receiving nodes. In a highly power constrained scenario TDMA must be implemented to reduce these possibilities while incorporating the advantages presented by the CSMA algorithm. TDMA, however, is impossible without tight clock synchronization. Though CSMA requires clock synchronization or awareness to intelligently interpret the incoming data, its requirements on clock synchronization are much less stringent than the TDMA algorithm whose transmission and reception slots are derived from the synchronization. Lack of the same would result in two nodes transmitting at the same time, defeating the purpose of the protocol.

The computer clock is an electronic device that counts oscillations in an accurately-machined quartz crystal, at a particular frequency. These are essentially timers which count each oscillation of the crystal and increment/decrement a register value until the value overflows and a timer interrupt is generated. Calculations performed using these timers provide the computer with a notion of time, and are used to provide timestamps on events, and set up TDMA slots when used.

To develop synchronization schemes, there are two significant issues regarding the clocks that must be understood and taken into consideration. The first is the *phase offset*. The nodes in a WSN are turned on at different times. When two nodes attempt to communicate with each other their local clocks may have different values and this difference is referred to as the phase offset. The second problem is termed *clock skew*. Clock skew arises from the difference in the oscillator's expected and actual frequencies, and the maximum is usually reported by the manufacturer. Besides *inaccuracy*, clock skew is also attributed to the *frequency instability* of crystals, which causes the clock to speed up or slow down over time. *Short-term* instability is often caused by environmental factors such as temperature, and *long-term* instability is caused by more inherent problems such as oscillator aging [18]. Present day oscillators are accurate to one part in 10^4 to 10^6. A frequency deviation of simply 0.001% will cause a clock error of a second a day which necessitates the need for repeated clock synchronization. Before delving into existing clock synchronization techniques, we discuss criteria to judge the performance of proposed clock synchronization algorithms.

1.4.1 Evaluation Criterion for Clock Synchronization Protocols

Clock synchronization protocols differ immensely in their energy requirement, precision, transmission requirement, means of incorporation into the MAC protocol, and their ability to be generalized for large scaled networks. The choice of a synchronization method would thus be highly dependent on hardware availability, the application at hand, and the MAC protocol chosen to govern the network. None of the existing methods [16-22] clearly outweigh the others in every metric evaluated and hence, present the user with tradeoffs to consider specific to the intended application. The following are essential metrics to evaluate before adopting or developing a clock synchronization scheme.

1.4.1.1 Synchronization Precision

The synchronization precision required from the protocol varies on the application, MAC protocol, and desired reporting time (frame length in TDMA). This metric reflects the maximum deviation in the notion of time after synchronization has been performed, and places bounds on the slot size, and slot length that can be achieved without interference, and on the tightness of data fusion algorithms used at the sinks. During event comparison at a sink, the precision of the time-stamps will determine the efficacy with which the sink can resolve the order of events. For example, applications which require the calculation of velocity and direction of motion of an object might require extremely high synchronization accuracy. Precisions varying from 1.85μs [16] to 3ms [17] have been reported by popular clock synchronization publications. These precisions are achieving under varying topologies and resources in WSNs, and should not be compared directly, unless reported for extremely similar conditions.

1.4.1.2 Piggybacking

Piggybacking is a term used to describe the process of combining synchronization messages with data messages sent amongst nodes. Most synchronization schemes involve synchronization phases which reduce bandwidth, add control overhead, and increase the data storage requirements at destination nodes. The presence of piggybacking in the algorithm alleviates the communication demands on the network. Though synchronization is a necessity, piggybacking renders a scheme more valuable, since the primary function of the network is to sense and communicate data.

1.4.1.3 Synchronization Message Length

Since each byte transmitted by the network consumes significant amounts of energy, synchronization packets that require complex messages are detrimental to the low-power requirements of the network, given that synchronization must continuously take place. This introduces the need for intelligent synchronization packets, which not only save power, but also reduce the convergence time.

1.4.1.4 Convergence Time

Convergence time is the total time required to synchronize the network. Protocols requiring a large number of message exchanges per synchronization result in longer convergence times. Reducing this convergence time is directly related to decreasing bandwidth required and power consumption at each node. In addition, long convergence times may hinder the sensor nodes' sensing capabilities, or the capability to report the sensing event in the response time desired. Convergence time and message length are of less criticality in protocols that propose infrequent synchronization [17, 21].

1.4.1.5 Complexity

Power and hardware constraints at each node coupled with the continuous environment monitoring tasks make highly complex synchronization schemes undesirable for several applications. A survey of literature reveals synchronization schemes tailored to multi-hop routing [19, 21, 22], to be of higher complexity. The complexity, hence, should be contingent to the size of the network, and the precision of synchronization required, while being well within the computational ability of the motes. Complexity that limits the sensing ability of nodes in crucial applications could defeat the purpose of using wireless sensor networks to approach the problem.

1.4.1.6 Network size and Scalability

Schemes to extend local synchronization techniques to the entire network are essential in WSNs. In addition, empirical evaluations of devised synchronization protocols on actual sensor networks are essential to prove their feasibility, performance, and scalability. Several algorithms in literature have not been applied to self-designed or existing sensor platforms. Simulations, though indicative of the scalability of the protocol, may not reveal hardware dependent challenges faced while implementing the scheme. Although several authors have not measured scalability in their publications, it is one of the top priorities in any clock synchronization protocol. The desire to incorporate tens-of-thousands of low cost nodes in a network to realize the specific application makes it imperative to find techniques to scale the concept to much larger networks. For example, innovative small-scale (*hundreds*) peer-to-peer topologies must outline or subsequently devise strategies to extend their synchronization to distant network nodes through techniques such as clustering, or multi-hop cascading. Several innovative MAC protocols result from scalability solutions offered by separate authors to existing small-scale synchronization schemes [23].

1.4.1.7 Compatibility with Sleep Mode

Besides collision avoidance, the best solution to the low-power needs of the network seems to be *sleep modes* for the nodes: periods where the devise may recede into a low power mode by turning off its radio, to prevent idle listening [10]. To achieve this, clock synchronization must be acquired before the sleep time and be tight enough for the node to remain synchronized when it “wakes up”. For less accurate synchronization schemes, mechanisms to immediately get resynchronized on waking up must be developed. Synchronization techniques which are accurate but extremely short-lived would eliminate the possibility of sleep cycles hence removing an excellent power saving solution.

Besides the above mentioned criteria, there are other criteria to be considered which may be tied to the existing metrics. Overall energy efficiency reflects the protocol’s success in balancing the tradeoffs of each separate quantifying metric. Overall accuracy is important in networks which require a timestamp with respect to an external standard of time. A rather infrequently studied issue is that of fault tolerance [20, 22], which represents the protocol’s handling of erroneous messages owing to the limited connectivity and interference in the network. Addressing message loss is an important issue as significant overheads are involved in retransmitting, and nodes might lose synchronization in the face of such loss.

1.4.2 Clock Synchronization Protocols

Traditional clock synchronization methods [25] used in wired networks fail to satisfy the stringent requirements of WSNs. Robust techniques with lower message exchange are required to adhere to requirements stated in section 1.4.1. This section discusses and analyzes existing clock synchronization schemes for wireless sensor networks, and draws attention to important issues overlooked in popular clock synchronization surveys [25].

1.4.2.1 Continuous Clock Synchronization

The continuous clock synchronization method was presented in a sequence of two publications by Mock et al. [35, 18]. The authors stressed the need to comply with the IEEE 802.11 standard, commonly accepted for wireless local area networks (WLANs). The standard outlines a master/slave configuration, using special nodes referred to as access points, to alternate CSMA based MAC periods, and a contention free synchronization period. In the synchronization period the access point, at a time t_2, transmits a "beacon frame" which includes a time-stamp of the master's local clock, t_1, which is received by the slaves at t_3, who adjust their clocks at t_4. In their work [35], Mock et al. attempt to increase the precision of the IEEE 802.11 clock synchronization protocol by revisiting a concept presented in the early nineties [41, 42] demonstrating the broadcast property of a wireless communication medium. This property implies that when two receivers receive the same message, the reception is tight, and they receive it at approximately the same time. Exploiting this observation, they propose a method where the master prepares a message at t_1, and transmits it at t_2. Each slave, and the master, receive this beacon at t_3, and take a local stamp of the event at t_s. The concept of the master monitoring the reception of its own broadcast allows the master to send a second message containing its time-stamp, t_s, to the slaves. Upon reception, the slaves compute the difference of their own time-stamp to that received from the master, and adjust their clocks at a time, t_6. The above explanation, and the time graphs in figure 1.4, demonstrate the reduction of the time-critical path of t_1 to t_4 in the IEEE standard, to t_3 to t_s, in Mock et al's method. To alleviate the waste of bandwidth in sending two messages, the method regards the second time-stamp message as a new indication message, hence incorporating the concept of piggybacking master time-stamps for the previous message onto the new indication message.

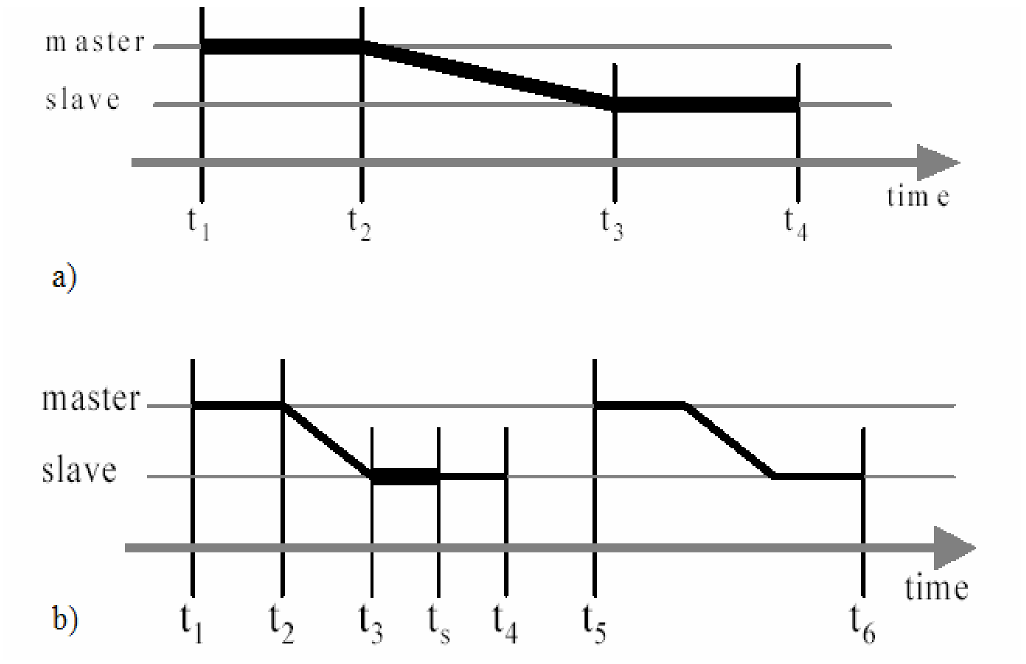

Figure 1.4 [35] a) Time critical path (t_1 to t_4) in the IEEE 802.11 clock synchronization. b) Reduced time critical path (t_3 to t_s) in Mock et al.'s scheme.

The author's also acknowledge that the need to receive two consecutive messages reduces the fault-tolerance of the method, a major concern in WSNs. To improve the fault-tolerance, the i^{th} indication message contains the time-stamps for the last n synchronization messages, tolerating up to $n-1$ consecutive message losses. To illustrate this concept they assume

sm_i: the i^{th} synchronization message

tm_i: the master time-stamp for the i^{th} message

cm_i: the slave time-stamp for the i^{th} message

and, sm_i contains the values of tm_{i-n}, tm_{i-n+1}, …., tm_{i-1}. If the last message received by a slave was sm_j, where $i - j < n + 1$, the slave looks up the value for tm_j in the message, and adjusts its clock based on the value $cm_j - tm_j$.

Besides considering the fault-tolerance of their proposed method, Mock et al. brought attention to the issue of time discontinuity while correcting clocks [18]. They argue that instantaneous clock adjustment leads to faulty local interval measurement, and propose a method of clock correction over a synchronization interval. The mechanism is best understood studying their time adjustment graph shown in figure 1.5. They introduce the notion of a virtual clock, which is a function of the slave's physical clock, and attempts to mimic the master's clock. The physical clock is based on the slave's oscillator and cannot be adjusted. For reasons of simplicity, this transform is assumed to be linear. The figure contains the following symbols

MC: the master's calculated clock

SC: the slave's virtual clock

SC_{i-1}: the uncorrected virtual clock

SC^*_i: the virtual clock during correction

SC'_i: the virtual clock after correction

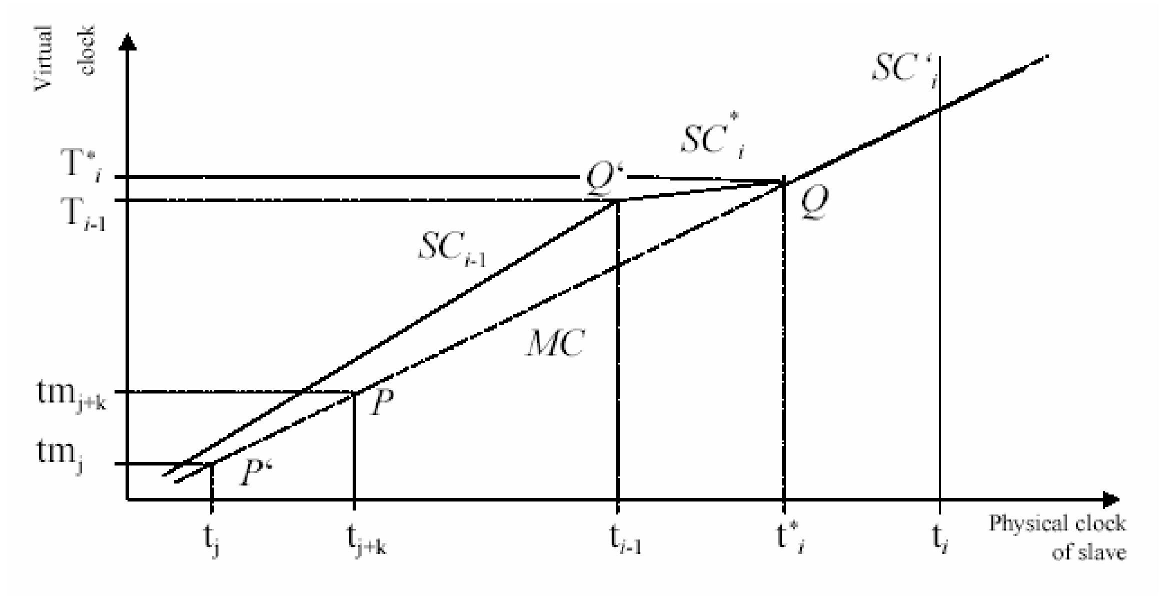

Figure 1.5 [18] Time adjustment graph for continuous clock correction

The difference in value of the corresponding points on MC and SC_{i-1} represents the phase offset, and the different slopes represent the clock skew causing the two clocks to increment at different rates. The goal is to correct SC_{i-1}, using SC^*_i, to SC'_i, a line with the same slope as MC, and starting at a point which lies on both MC and SC^*_i. Since it takes two points to completely describe a straight line, MC is calculated from points P' (t_j, tm_j) and P (t_{j+k}, tm_{j+k}), which represent the jth and j+kth time-stamps from the master, respectively, in terms of the slave's physical clock on the x-axis, and virtual clock on the y-axis. Q' (t_{i-1}, T_{i-1}) and Q (t^*_i, T^*_i) denote the start and end points of the correction interval, where *i-1* is the last received synchronization point, and $t^*_i = t_{i-1} + b_{correct}$, where $b_{correct} > 0$ is a prior known protocol parameter. From the figure, we can see that $T_{i-1} = SC_{i-1}(t_{i-1})$, and $T^*_i = MC(t^*_i)$, since Q' lies on SC_{i-1} and Q lies on MC. After finding the transform for correction, SC^*_i, using Q' and Q, we notice that the corrected transform

SC'_{i-1} is the same line as MC, and its equation is derived from the same two points, P and P' used to determine MC.

Mock et al have implemented their proposed protocol using WLAN Network Interface Cards (NIC). They report a synchronization precision of 150µs compared to the 1000µs precision of their implementation of the IEEE standard. Being one of the first comprehensive clock synchronization proposals, their precision of 150µs may be a conservative estimate, given that implementation on wireless motes with transmission and reception based on interrupts, would result in tighter synchronization. A literature survey did not reveal Mock et al's method being incorporated on existing WSN platforms. Though not explicitly outlined in their publications, their technique of requiring the master to receive its own transmission requires the capability of transmitting and receiving at the same time. Most wireless sensor motes comprise streamlined transceivers, which can only either transmit or receive at a time. Their technique, though very effective, would necessitate separate receivers and transmitters on each mote, which may not be practical without significant advances in radio frequency technology with respect to power and size. Their continuous clock synchronization scheme though more complex than instantaneous correction, brings to attention the issues of time-discontinuity, and offers developers a method of achieving interval correction irrespective of the synchronization method chosen. The need for the same may be removed by devising scheduling and rescheduling techniques for events which lie in the correction interval.

1.4.2.2 Reference Broadcast Synchronization (RBS)

The RBS method [16] is closely related to Mock et al's scheme [35, 17] since it takes advantage of the broadcast property of the wireless medium. It merits discussion owing to its fundamental property of achieving synchronization amongst a set of receivers, rather than between sender and receiver, its extension of the protocol across broadcast domains, and successful implementation of RBS by three research groups. Their publication discusses the non-determinism in synchronization characterized by Kopetz and Schwabl [36]. Also referred to as message latency, this non-determinism comprises, the *send time*; time spent by the sender to prepare the synchronization message, the *access time*; time taken by the sender to access the transmit channel, *propagation time*; time taken for the message to reach the receivers once it leaves the sender, and the *receive time*; time taken by the receiver's network interface to receive the message from the channel. As in Mock et al's scheme the first two components are removed by using the broadcast property of the medium. The authors further argue that the effective propagation time is 0 since electromagnetic waves travel through air a speed close to c, or $3 \text{ X } 10^8$ m/s, and through copper wire at 2/3rd c. Though propagation delay dominates the delay in Wide Area Networks (WANs), in WSNs, where nodes are of the order of 10 meters or 30 feet apart from each other, this delay is at most tens of nanoseconds, which is inconsequential to the μ-second-scale error budget. Recognizing the *receive time* to be the only remaining source of error, the authors report a series of experiments, using the Berkeley motes [8], to characterize this error. The maximum phase error recorded was 53.4μs which the authors correlate to the 52μs bit time of the motes. The bit time is the time required to receive one bit of information from the channel, and is 52μs at 19,200 baud. Another important result was the Gaussian nature of the phase offset distribution shown in figure 1.6, which motivated them to send multiple broadcast packets, and aided in realistic receiver modeling in numeric analyses.

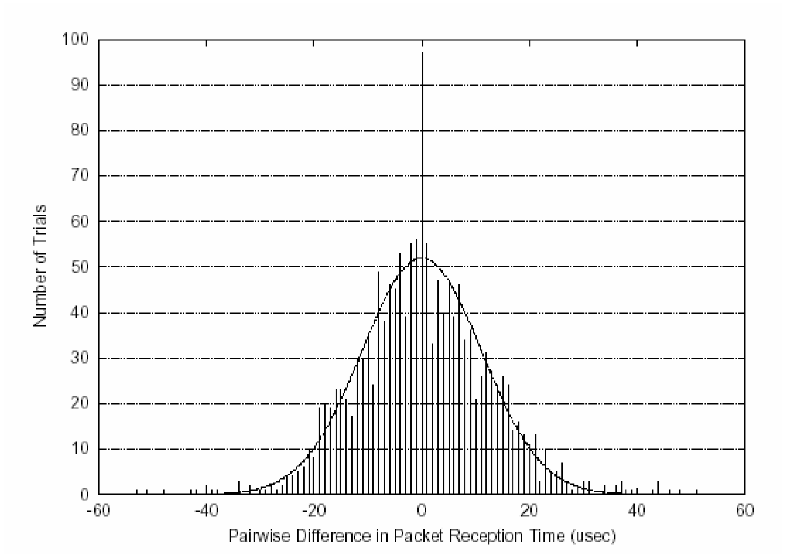

Figure 1.6 [16] Histogram for inter-receiver phase offsets for 1478 packets

Elson et al., next, focused on the two significant sources of desynchronization of clocks, phase offset, and clock skew. To remove the phase offset they propose the following algorithm:

1. A sender broadcasts *m* synchronization packets
2. The *n* receivers record the time of arrival according to their local clocks
3. The receivers exchange these recorded arrival times and each receiver *i* computes its phase offset to any other receiver *j* as the average of their phase offsets for each reference packet

$$Offset[i,j] = \frac{1}{m}\sum_{k=1}^{m} T_{j,k} - T_{i,k} \tag{1.2}$$

To estimate the clock skew, the synchronization scheme is modified to perform a *least-squares linear regression* through the phase offsets between two nodes instead of averaging them, as depicted in figure 1.7a. This scheme provides a fast method for finding a best fit line through the phase error observations, with the slope of the line representing the clock skew, and the intercept revealing the initial phase offset. Short-term frequency stability is an assumption in the method since fitting a line to the data implicitly assumes that the frequency is stable. An important point to note is that the RBS method does not involve clock correction. Instead, a table of phase offsets and clock skews of the other receivers is calculated and maintained at each node. Elson et al. have also proposed *post-facto synchronization* [37] which is a scheme of allowing the nodes to normally stay in low power mode with unsynchronized clocks until an event of interest occurs. This event triggers the synchronization protocol.

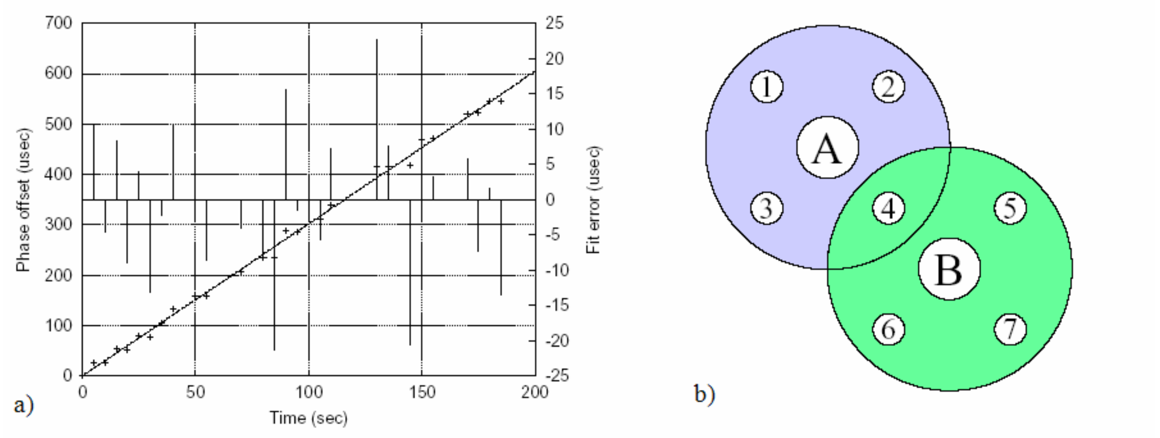

Figure 1.7 [16] a) Least squares linear regression through phase offsets between two receivers b) Network topology with A, B as sync senders, and 1-7 receivers.

The final contribution made by Elson et al. was extending the RBS protocol to operate outside single broadcast domains in a multi-hop environment. To illustrate their technique they refer to the scenario depicted in figure 1.7b, where A and B both send synchronization pulses, and receiver *R4* is the only node that hears both these broadcasts. Assuming that two events E_1 and E_7 occur at receivers R_1 and R_7, respectively, to compare the times of occurrence the following steps are followed:

1. Receivers R_1 and R_7 observe events $E_1(R_1)$ and $E_7(R_7)$ which are recorded in terms of their own local clocks
2. Using A's broadcast R_4 calculates the best fit line needed to convert clock values from R_1 to R_4. R_4 performs this conversion after exchanging reception times of A's messages with R_1, obtaining $E_1(R_4)$ from $E_1(R_1)$
3. Similarly, B's broadcasts are used to convert $E_1(R_4)$ to $E_1(R_7)$
4. The time difference in the occurrence of the events is $E_1(R_7) – E_7(R_7)$

The authors suggest weighted shortest-path algorithms such as Dijkstra's algorithm to find the best conversion path, with the weight of each edge being the quality of conversion represented by the calculated residual synchronization error.

Elson et al. implemented their algorithm on the Berkeley Motes, achieving a precision of 11µs at 19,200 baud. On commodity hardware, Compaq IPAQ's using an 11Mbit/sec 802.11 network, the precision was improved to 6.29 ± 6.45µs. Using kernel timestamps on the same platform they report a precision of 1.85 ± 1.28µs. They also report that the multi-hop precision error generally seems to follow an estimate of $\sigma \sqrt{n}$ for n hops and an average per-hop error of σ. Besides a localization service implemented on the Berkeley motes using RBS, Merill et al. report Sensoria Corporation's use of RBS in their distributed autonomous position location system [38], and Wang et al. report using the RBS daemon in IPAQs for a distributed beam-forming application [39]. The RBS protocol achieves an excellent precision, offering developers a scheme of achieving the same when required. It is achieved, though, at the cost of multiple broadcasts, and message exchanges between receivers, consuming power, and should be used only if a precision of the order of tens of μseconds is necessary. The biggest drawback of the method is the sender remaining desynchronized from the rest of the nodes, preventing these pulse sending nodes from contributing to sensing and communication functions in the network. For WSN implementations, the 11μs precision should be used for comparison with other implemented synchronization protocols. Also, the absence of clock correction seems to render the protocol more powerful for event time-stamping rather than setting up TDMA slots, and can be used in TDMA networks for increased time-stamp precision.

1.4.2.3 Time-diffusion Synchronization Protocol (TDP)

The time-diffusion synchronization protocol (TDP) [19] proposed by W. Su and I. Akyildiz is an important synchronization scheme to study since it introduces a novel mechanism of achieving a network-wide equilibrium time by diffusing timing information through a tree-like structure formed by the wireless sensor motes. Their extremely detailed protocol achieves synchronization during TDP active phases which are followed by TDP inactive phases during which sensor and network operations are performed. The authors help visualize this using figure 1.8. Each active phase involves the election/reelection of master/diffused leader node procedure (ERP), during which nodes self-determine whether to become a master and initiate the time diffusion process, or qualify as diffused leaders to propagate the master's timing information through the tree. The elected masters initiate the peer evaluation procedure (PEP) to accumulate and compute critical timing information of its neighbors. This is followed by the time diffusion procedure (TP) where the master nodes diffuse the timing information every δ seconds (round interval) for a duration of τ seconds (round length). Neighboring nodes receive this information, use the ERP to determine if they qualify as diffused leaders, and use the time adjustment algorithm (TAA) to adjust their clocks after waiting for δ seconds.

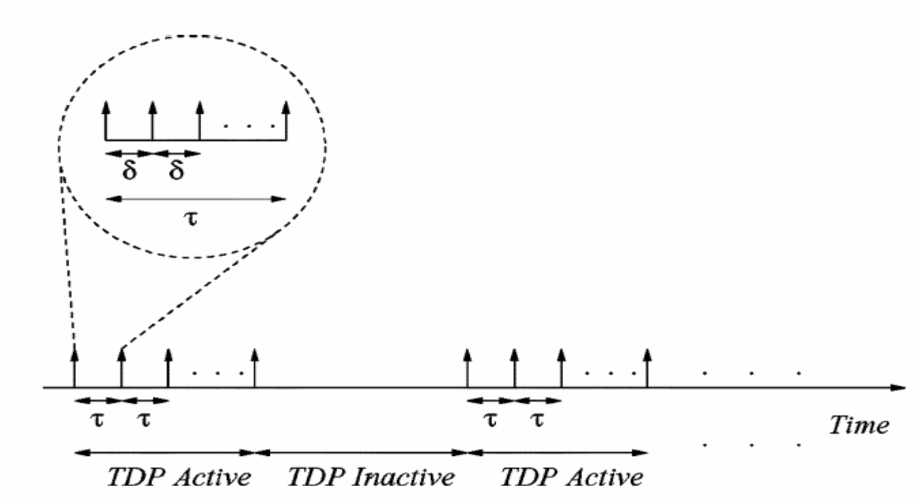

Figure 1.8 [19] TDP active/inactive schedule

The PEP is aimed at allowing the nodes in a neighborhood to evaluate the quality of their clocks using a variance method. The PEP is realized using the following steps:

1. The elected master nodes broadcast η time-stamped *SCAN* messages.
2. The nodes which receive a master's messages, calculate a variance $\sigma^2(\iota)$ of their local clock from that of the master using the following equation:

$$\sigma^2(\iota) = \frac{1}{2\iota^2(N-2)} \sum_{g=1}^{N-2} \left(x_{g+2} - 2x_{g+1} + x_g \right)^2 \tag{1.3}$$

where ι is the time difference between two time deviation measurements, N is the total number of time deviation measurements i.e. the number of *SCAN* messages received by the node, and x is the calculated time deviation.

3. These calculated variances $\sigma^2(\iota)$ are communicated to the master in the *REPLY* message.
4. The master node calculates an outlier ratio γ_{yz}, which is a measure of the deviation of the local clock of the sensor node z, from the clock of the master node y, with respect to the other neighborhood nodes using the following equation:

$$\gamma_{yz} = \left| \frac{\sigma_{yz}^2(\iota) - \sigma_{avg}^2(\iota)}{\sigma_{avg}^2(\iota)} \right| \text{ with } \sigma_{avg}^2(\iota) = \frac{\sum_{z=1}^{M} \sigma_{yz}^2(\iota)}{M} \text{ and } \sigma_{avg}(\iota) = \frac{\sum_{z=1}^{M} \sqrt{\sigma_{yz}^2(\iota)}}{M} \tag{1.4}$$

where, $\sigma_{avg}^2(\iota)$ is the average neighborhood variance, M the total number of *REPLY* messages received by the master, and $\sigma_{yz}^2(\iota)$ is the variance between nodes y and z calculated using the equation 1.3.

5. The master then sends back the outlier ratios γ_{yz}, and the average deviation $\sigma_{avg}(\iota)$ in *RESULT* messages.

The PEP provides the neighborhood nodes with information to evaluate the quality of their clocks with respect to the master and their neighbors by the ERP which will be discussed in a later paragraph. The TP, which diffuses timing information from the master nodes to the network nodes, merits reference to Su et al.'s publication [19] for thorough understanding, and is summarized in this paragraph. After receiving the messages, the nodes use the TAA to adjust their clock. The timing information handshake comprises the following fields:

1. Master node local ID (*M-LID*): ID of the master node initiating the diffusion
2. Source *LID*: ID of the node currently broadcasting the information
3. Value *n*: the number of levels or hops the message is to be diffused
4. Time $t_{M,i}$: The diffused time of the master, *M*, to be synchronized to in round *i*, calculated by the master using

$$t_{M,i} = t_{M,i} + \frac{\sum_{m=1}^{L} \Delta j_m}{2L} + \delta \tag{1.5}$$

   where δ is the amount of time the neighboring nodes wait to adjust their clocks, *L* is the total number of acknowledgment (ACK) messages received by the master in response to the timing information, $\Delta j_m = t_{1,m} - t_0$, where $t_{1,m}$ is the time at which the ACK from the m^{th} node is received by the master, t_0, the broadcast time of the initiated timing message, and Δj_m, the roundtrip time for the m^{th} node.

5. Value $\beta_{M,k}$: $\beta_{M,k} = \beta_{M,k-1} + \alpha$, where α is the standard deviation of the roundtrip times Δj_m, and $\beta_{M,k}$ the accumulated standard deviation is a function of the number of hops *k* from the master, and is used to estimate the quality of the diffused time in the TAA.

The master diffuses a new timing message every δ seconds, for a duration of τ seconds to accommodate clock variations, and mobile nodes. The diffused leaders perform the same TP to propagate the timing information. At each sensor node, the TAA uses the $\beta_{M,k}$ diffused values from different masters to generate a weight, w_M, for each master. w_M is large if the master M is fewer hops away. A new time is calculated as a weighted sum of the diffused times. The clock is adjusted only if the computed correction is greater than the received neighborhood deviation $\sigma_{avg}(\iota)$.

A thorough understanding of the PEP and TP help in analyzing the Election/Reelection of Master/Diffused Leader Node Procedure (ERP) which governs whether nodes self-determine themselves to be master nodes or diffused leaders. The ERP is realized using the False Ticker Isolation Algorithm (FIA) and the Local Distribution Algorithm (LDA). Using the FIA, a node checks if its received outlier ratio γ_{yz}, is greater than a threshold ϕ. If greater, the node regards itself an *outlier*, and disqualifies itself from being a diffused leader in the current τ period, and a master at the beginning of the next τ period. The LDA ensures that the TDP is power efficient. The masters are reelected every τ seconds, and diffused leaders, every δ seconds. Each node randomly selects a value λ between 0 and 1, and shifts this number by $(1-\zeta)$ where ζ is the ratio of the current energy level over the maximum allowed energy level, implying $\lambda = \lambda - (1-\zeta)$. If λ is greater than a threshold φ and the node is not a false ticker, the node qualifies as either a master or diffused leader when the nodes are being reelected. The formulation of the threshold φ discussed in detail in [19].

The TDP, though not implemented on a sensor platform, appears to be an extremely thorough mechanism of achieving a network-wide equilibrium time within a certain tolerance, especially for mutlihop and mobile WSNs. The authors provide simulation results to validate the efficacy of their protocol. The scheme, however, before adoption, requires designers to study trade-offs of power, complexity, and time for the intended application. The need to define and study multiple thresholds, and timing parameters, while ensuring enough time to perform sensor operations, seems to be a rather challenging task. Also, if the PEP and TP are achieved using CSMA, collisions can make this synchronization scheme highly power consuming.

1.4.2.4 Time Synchronization in Ad-hoc Networks

This synchronization protocol was devised by Romer [17] to facilitate time synchronization in ad-hoc communication networks. Ad-hoc communication networks are characterized by mobile computing devices which communicate with each other when they enter each other's communication range. This is achieved via a bi-directional communication link and node synchronization can be achieved at these times. These links are usually very short range and they last a short time, given the mobility of the devices. Romer argues that two nodes which do not enter each other's communication range can communicate via store and forward techniques. In this scheme, an intermediate node receives the sender's message, stores it for a while, and transmits it to the intended receiver when they enter each other's communication range. Romer's method is primarily based on recognizing two main differences between synchronization conditions in traditional networks and ad-hoc networks. While traditional networks depend on accurate delay estimation [16, 35], this is not feasible in ad-hoc networks, as an arbitrarily large time may be required for the message to reach the receiver. Secondly, periodic message exchanges [19] are not possible to maintain synchronization since it is impossible to achieve synchronization between every pair of nodes in the network.

The driving concept behind the algorithm is determining lower and upper bounds for the actual time elapsed between creation of a timestamp at the source node, and the reception of this timestamp at the destination node. The bounds are necessary because in ad-hoc networks the time transformations cannot be achieved with high precision. The receiver, then, translates these bounds in terms of its local clock, and subtracts the resulting values from the time of arrival of the message, thus obtaining the upper and lower bounds of the received timestamps in terms of its own clock. Assuming that computer clocks have a maximum clock drift of ρ, Romer obtains an inequality relating time differences Δt to clock differences ΔC using the maximum clock drift ρ:

$$1-\rho \leq \frac{\Delta C}{\Delta t} \leq 1+\rho \tag{1.6}$$

For an ideal hardware clock, $\frac{dC}{dt}=1$. Present day hardware can achieve ρ values of approximately 10^{-6}. The inequality can modified to $\frac{\Delta C}{1+\rho} \leq \Delta t \leq \frac{\Delta C}{1-\rho}$, implying that the computer clock difference of ΔC is bounded by the interval $\left[(1-\rho)\Delta t,(1+\rho)\Delta t\right]$, and conversely, the time difference Δt is bounded by the interval $\left[\frac{\Delta C}{1+\rho},\frac{\Delta C}{1-\rho}\right]$. These derivations lead to equations for converting a clock time difference of ΔC_i from the local time of node *i,* with ρ_i, to the local time of another node *j*, with ρ_j. First ΔC_i is estimated by the real-time interval on node i $\left[\frac{\Delta C}{1+\rho_i},\frac{\Delta C}{1-\rho_i}\right]$. Subsequently, it is converted to the computer time interval $\left[\frac{\Delta C\left(1-\rho_j\right)}{1+\rho_i},\frac{\Delta C\left(1+\rho_j\right)}{1-\rho_i}\right]$ with respect to the local time on node j.

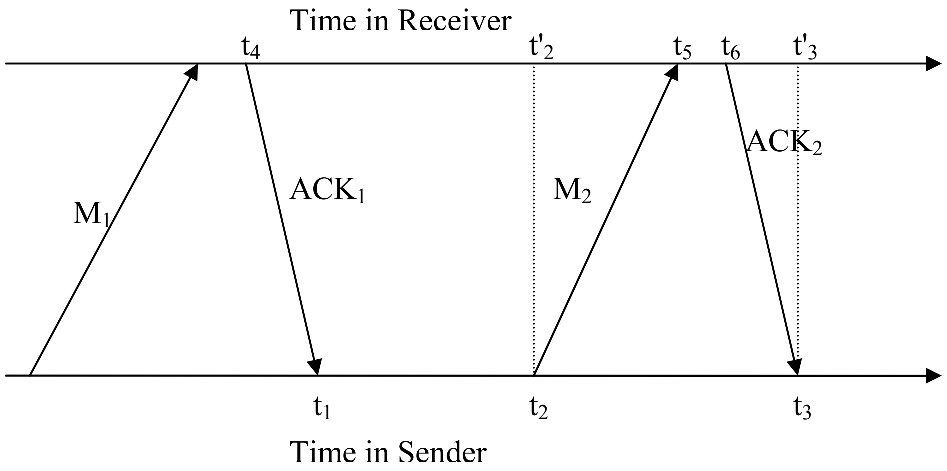

Figure 1.9 Message delay estimation using Romer's method [17]

Romer's algorithm focuses on estimating the delay d of sending messages between two nodes. This is accomplished by one node sending a message, to which the other responds with an ACK. The time difference between sending the message and receiving the ACK is the round-trip time (*rtt*). Once the sender receives the ACK, it is aware of the *rtt*, but must use a second message to communicate this value to the receiver. This second message can also be used by the receiver to compute a second *rtt*. The synchronization, therefore, is achieved with a 100% message overhead, since it necessitates two messages. Figure 1.9 is a modified version of a diagram used by Romer to explain his algorithm [17], with a couple of additional time labels to illustrate the concept, and explain some nuances omitted in [17]. The figure shows two consecutive messages exchanged between two nodes. To estimate the delay *d* for message *M₂*, we recognize $d = t_5 - t'_2$. Recognizing the non-determinism in calculating t'_2, Romer states the lower bound of *d* to be 0, and calculates the upper bound using the above estimation resulting in the inequality:

$$0 \le d \le (t_3 - t_2) - (t_6 - t_5)\frac{1-\rho_s}{1+\rho_r} \tag{1.7}$$

where $(t_3 - t_2)$ is the round-trip time of M_2, $(t_6 - t_5)$ is the idle or storage time of M_2 within the receiver in terms of the receiver's clock, and ρ_s and ρ_r are the maximum clock drifts for the sender and receiver, respectively. Also, Romer refers to the upper bound of *d* as the *rtt*, though he intends it to be (*rtt – idle time).* Though this includes the time $t'_3 - t_6$, it serves as a true upper bound for the delay. A second estimation of the delay in terms of the receiver's clock makes use of two consecutive message transfers:

$$0 \le d \le (t_5 - t_4) - (t_2 - t_1)\frac{1-\rho_r}{1+\rho_s} \tag{1.8}$$

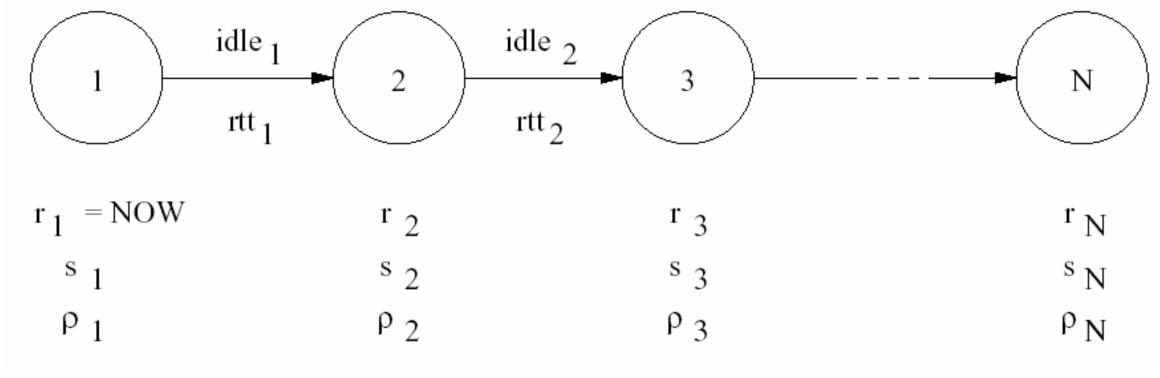

Figure 1.10 Message flow graph for ad-hoc network [17]

After defining the upper and lower bounds of the delay, Romer outlines a scheme to generate event time-stamps, and defines rules for time-stamp comparison and translation. An event *E*, occurring at an exact clock time $C_i(E)$, is recorded by a time-stamp $S_i(E)$, represented by the interval $\left[C_{i,l}(E), C_{i,r}(E)\right]$. This interval, with a lower and upper bound for the event, indicates that time of occurrence of *E* is only an estimation because of the inaccuracy of the local clock.

For comparison of these time-stamps, the scenario in figure 1.10 represents device 1 sensing an event, recording the time of the event and communicating this information to the devices 2, 3, ……, *N* along the path shown. Each node *i* records r_i, the time at which the message containing the time-stamp is received s_i, the time at which the time-stamp is sent from node *i*, and the maximum clock drift ρ_i, which is a device parameter. Each connecting edge is represented by rtt_i, the round-trip time for sending the message from node *i*, in terms of the clock of the receiving node, and $idle_i$, the idle time elapsed in node *i* after sending the last message in terms of the sender's clock. r_i, s_i, and $idle_i$ are contained in the time-stamp message sent over the edge. Since the first node does not receive any messages, it is assumed that the event is recorded at time r_1. We then proceed to derive the upper and lower bounds presented by Romer for the event at each node. Adding the notations *UB* for the Upper Bound, *LB* for the Lower Bound, $store_i$ for the storage time at node *i*, and $delay_i$ for the delay from node *i*, we derive:

Node 1: $[r_1, r_1]$

Node 2: $[r_2 - UB(store_1) - UB(delay_1),\ r_2 - LB(store_1) - LB(delay_1)]$ (1.9)

where $UB(store_1) = (s_1 - r_1)\dfrac{1+\rho_2}{1-\rho_1}$ and $LB(store_1) = (s_1 - r_1)\dfrac{1-\rho_2}{1+\rho_1}$

$$UB(delay_1) = rtt_1 - idle_1 \frac{1-\rho_2}{1+\rho_1} \text{ and } LB(delay_1) = 0$$

$$\text{Node N: } \left[r_n - \sum_{k=1}^{N-1} UB(store_k) - \sum_{k=1}^{N-1} UB(delay_k), r_n - \sum_{k=1}^{N-1} LB(store_k) \right] \tag{1.10}$$

As indicated by Romer, this method is tailored to achieve synchronization in sparse ad-hoc networks. A time-stamp lasting up to 5 hops, and an age of up to 500 seconds, is reported to have an accuracy of no less than 3ms. This 3ms precision should not be directly compared with achieved precisions of broadcast mechanisms in dense sensor networks, as it is achieved in a sparse ad-hoc network, even after several hops and a large elapsed time. Romer's method provides an excellent synchronization scheme in conditions where the RBS [16] and TDP [19] might not be applicable. The second method of delay estimation in terms of the receiver's clock has the disadvantage of potentially large *rtt* and idle time, introducing errors due to drift. In addition, timing information has to be maintained for older messages since the equation spans two message exchanges. This may be undesirable if the nodes are in contact with several other nodes in the network. Finally, we notice in the derivation of *N*-hops, communicating r_i, s_i, rtt_i and $idle_i$ for each intermediate node can cause the message to become increasingly large, in which case the protocol starts utilizing higher bandwidth, and the delays are calculated for messages of significantly different length.

1.4.2.5 Other Synchronization Protocols

Other synchronization protocols include the Probabilistic clock synchronization service [21] which provides probabilistic accuracy bounds on the accuracy of the RBS clock synchronization scheme. The Asynchronous Diffusion Protocol [22] proposed by Li and Rus defines a rate-based diffusion protocol that is achieved by flooding neighboring nodes with clock values to reach a consensus to adjust all clocks to. These and several other synchronization schemes are discussed in clock synchronization surveys [24-26].

The above discussion reveals that the choice of a synchronization protocol is highly dependent on the application, hardware resources, and network topology. The continuous clock synchronization protocol [35] is more suited to WLANs equipped with NICs which can transmit and receive simultaneously. The absence of this feature in sensor motes renders the scheme unsuitable to WSNs. However, the method of interval correction [18] can be incorporated if time discontinuity cannot be resolved. The RBS [16] protocol seems most suitable to contention based applications with high precision requirements for event time-stamping. Simulations predict that the TDP [19] can achieve excellent network synchronization in large networks which can afford the computational complexity of the protocol. Romer's method [17] is specific to sparse ad-hoc networks. For example, technological advancements have led to the development of cell-phone sized wireless devices [40] equipped with Inertial Navigation Units (INUs), to track first responders within buildings where GPS is unavailable. These devices use high power transmissions to communicate their calculated position with a base station, but may also be equipped with short-range bluetooth modules. While outdoors, they obtain absolute timing information from the GPS modules. Indoors, when two responders are within bluetooth communication range, they can exchange time-stamped location information to ensure their computed locations are within approximately 10 meters of each other, as a correction mechanism. This time-stamped information can be communicated using Romer's algorithm to other personnel in the building to aid in a search and rescue scenario.

1.5 Motivation and Outline

The Smart Dust Project is aimed at developing low-cost, low power wireless sensor nodes of the size of dust particles, to form large scale networks. These dust sized nodes are intended to independently form a network among themselves to solve a particular sensing task. A study of existing medium access control (MAC) protocols suggests that schemes solely based on CSMA, are usually easier to implement [13], but less power efficient than TDMA schemes. TDMA, though highly power efficient, presents challenges of developing schemes to physically implement the slots on sensor nodes, and deriving these slots after achieving tight synchronization in the network. Most proposed TDMA schemes [11, 14] assume clock synchronization and do not address the issue. In addition, they seem to neglect hardware issues which may arise while implementing these algorithms on existing sensor platforms. Furthermore, though some of the clock synchronization schemes discussed in section 1.4.2 achieve excellent synchronization [16, 19], they do not particularly fit into a TDMA framework. Motivated by these gaps at various levels in the design requirements of a complete protocol, the Smart Stone Protocol (SSP) has been developed. The protocol achieves rapid synchronization to form TDMA slots, synchronizes receivers within 50μs of each other, and achieves and maintains synchronization within a cluster without sending any synchronization bytes. The sender is not left unsynchronized and a novel scheme is used to identify the closest comparable times on the sender and receiver, eliminating/reducing the non-deterministic *send/access* delays. The protocol is tightly related to events that occur in the mote hardware, but is designed to operate on extremely constrained wireless sensor motes. To test and validate the protocol, Smart Stones have been custom designed using commercial off-the-shelf (COTS) components, and the SSP has been successfully demonstrated on the Smart Stone Network. Finally, the SSP has been extended to perform sensing operations and data communication within the network. An acoustic vibration sensing application has been tested on the Smart Stone Network, sensing the environment with MEMs microphones, performing local signal processing, communicating information within TDMA slots, and displaying results of data comparisons.

1.6 Chapter Organization

Chapter 2 is a discussion of the Smart Stones (wireless sensor motes of the order of inches) developed to test the SSP. The chapter details the choice and functionality of components, circuit design, Printed Circuit Board (PCB) layout considerations and techniques, manufacture and population of the PCBs, and the hardware blocks that govern the performance of all proposed algorithms.

Chapter 3 is a step-wise discussion of the evolution of the innovative TDMA algorithm used on the Smart Stones to achieve network synchronization, setup TDMA slots, and accommodate data transmission and fusion. The SSP is then measured against the limitations and metrics discussed in this chapter, and compared and contrasted with the discussed clock synchronization schemes and MAC protocols.

Chapter 4 concludes with proposed future work to develop the work in this thesis into a robust network protocol including clock synchronization, medium access control, and network-wide routing. In addition, features of existing clock synchronization, MAC, and routing protocols that may be incorporated into the SSP framework are highlighted.

CHAPTER 2

2. Wireless Sensor Nodes for Smart Dust

While Smart Dust is aimed at realizing wireless sensor nodes of the size of dust particles [28], this accomplishment is years away from materializing [29]. Smart Stones, wireless sensor nodes of size of the order of inches, are currently being developed to serve as test platforms for networking algorithms, and to serve in applications where their current size is satisfactory. Sensor nodes such as the MICA2 [8], and the TUTWSN node prototype [23] are built from commercial off-the-shelf (COTS) components. These platforms are crucial to test synchronization and communication schemes being developed. Well-designed smart stones can already be employed in habitat monitoring applications where dust size is not crucial. In addition to serving as a test network, a smart stone network necessitates the choice of particular hardware components whose features are used to customize the algorithm to the hardware, and also presents obstacles that cannot be easily perceived in theoretical simulations. Both the MICA2 and TUTWSN nodes, though remarkably compact, take advantage of rather specific microcontroller and transceiver technology, limiting the range of algorithms that can be tested on them. The networking algorithms designed as part of this thesis are intended to serve under stringent power restrictions. Algorithms were devised using the fewest possible instructions to facilitate easy design of Application Specific Integrated Circuits (ASICs) to replace the microcontroller. The use of ASICs is imperative in low-power applications to minimize hardware size, and perform the necessary functions in the fewest instruction cycles. Pottie and Kaiser [7] have pointed out that ASICs, which can clock at much lower speeds and use less numerical precision, consume several orders of magnitude less energy than digital signal processors (DSPs). Consequently, implementation and testing of these algorithms designed with ASICs in mind, required the development of streamlined wireless sensor nodes.

2.1 Smart Stone Hardware

Smart Stones were designed to implement Time Division Multiple Access (TDMA) communication between nodes with emphasis on software techniques to minimize power. Power considerations in hardware were given less importance acknowledging the parts chosen for the Smart Stones were not the final components desired in the Smart Nodes. To further develop the Smart Stone Network into a sensor network, provisions were made to accommodate sensor inputs into each node. The block diagram for the nodes is shown in figure 2.1. The Smart Stones comprise

- PIC 16F88 microcontroller from Microchip with a reset switch
- LINX TR-916-SC-P transceiver from LINX Technologies
- 20 MHz crystal
- 5V Voltage Regulator
- Reduced Height 916 MHz antenna
- Battery Input
- 5 display LED’s

- 3 pin sensor input

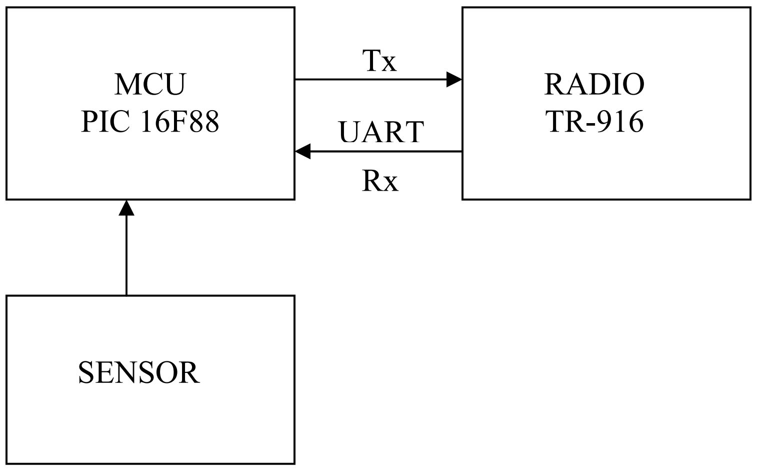

Figure 2.1 Block Diagram for a Smart Stone Node

2.1.1 Microcontroller (MCU)

The microcontroller used in the Smart Stones is the PIC16F88 [30] microcontroller from Microchip Technologies. The microcontroller is the heart of the signal processing, data aggregation, and data communication on the Smart Stones. All algorithms used are subject to limitations in hardware and the chosen microcontroller was required to support all processing and data input requirements. In particular, the transceiver required Universal Asynchronous Receive and Transmit (UART) ports to communicate with the MCU. Generic sensors output analog data which needed to be interfaced with the digital MCU. The PIC 16F88 has 7 analog channels with a built-in analog-to-digital (A/D) converter to furnish digital voltage levels to the MCU. The presence of this converter removed the need for a separate A/D converter chip, saving both complexity and board space. Other requirements included hardware timers on the MCU which are the basis of our proposed clock synchronization algorithm, and sufficient digital output pins to control LED's for display and testing, and controlling the transceiver. The 18-pin PIC 16F88 satisfied all these requirements providing UART channels, 7 10bit A/D conversion channels, built-in timers, and several digital output pins.

2.1.2 Transceiver and Antenna

The LINX TR-916-SC-P [31] transceiver is a module developed by LINX Technologies. Its communication frequency is 916 MHz and provides ease of switching between transmit and receive modes by toggling two input pins, TX enable and RX enable, both of which are connected to the microcontroller. This transceiver provides high transmission range which assists our development of peer to peer clock synchronization algorithms. The antenna incorporated into the Smart Stones is a reduced height, 916 MHz antenna to miniaturize the nodes while providing long range.

2.1.3 External Clock and Timers

The PIC 16F88 has an internal oscillator block which is capable of generating several clock signals. The main output is an 8 MHz clock source that can be directly used to drive the system clock. Post-scalars can be used to generate six different clock speeds between 31.25 KHz and 4 MHz. For our application, these clock sources are not sufficient since we need the microcontroller to run on a faster clock, producing faster timers, and more control over TDMA slot allocation. A 20 MHz external crystal is used to drive the microcontroller at a frequency of 5 MHz since the clock source drives the system at one-fourth the supplied frequency.

The device is also equipped with two 8-bit timers (*TIMER0* and *TIMER2*) with pre-scalars to slow the timer down. For the 8-bit timers there is a single register counting up, setting off an interrupt when there is an overflow. There is also an additional 16 bit timer which comprises two registers, one for the high byte, and one for the low byte.

2.1.4 Sensor Input

To test the Smart Stone nodes as a part of a sensor network, provisions were made to attach a sensor to the microcontroller's analog port. The nodes were designed for a preliminary application requiring acoustic MEMs microphones. Embedding the microphones in the board design would make the design more compact and remove possibilities of connection errors. However, initial tests using the microphone suggested that the microphone sensor would need to be delocalized from the board and be in contact with the ground to sense vibrations. Wire extensions make this delocalization possible. In addition, fixing the sensors would restrict the use of the motes to purely acoustic applications. Instead, the Smart Stones have a 3-pin connector providing power, ground, and an analog/digital port for the sensor reading. The desired sensor is then connected to the appropriate pins for the specific application.

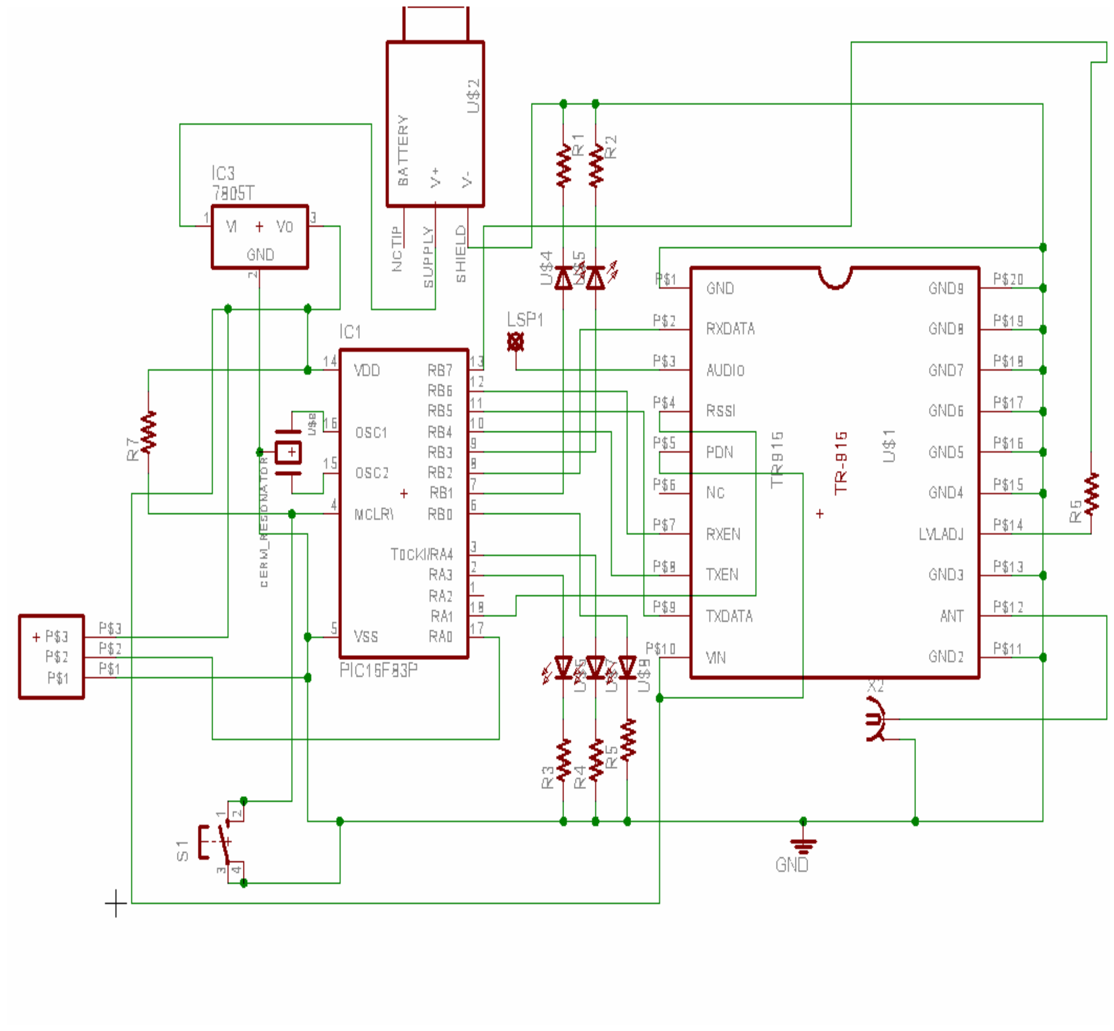

Figure 2.2 Schematic of Smart Stone PCB

2.2 Smart Stone Circuit Design

The block diagram in figure 2.1 depicts the control flow in the nodes. Each node is controlled by the MCU which is responsible for monitoring the sensor readings, processing these signals, and communicating them via the transceiver. Communication with the transceiver is achieved using the UART ports. Smart Stone nodes were designed to realize this functionality and first tested on a breadboard setup. Upon successful testing, the node was created on a Printed Circuit Board (PCB) design developed using Cadsoft *Eagle* [33] software. The schematic of the design is shown in figure 2.2. As shown in the figure, each Smart Stone is provided a battery input, ideally 4 AA alkaline batteries, providing around 6V to the board. This battery input is then passed through a 5V voltage regulator to supply exactly 5V to the microcontroller. The maximum Vdd voltage rating for the MCU is 5.5V and 5V is required to run a 20 MHz clock necessitating the use of a voltage regulator to protect the integrated circuits on the board. MCU ports are dedicated to control the transceiver. Two digital pins are connected to the transceiver's Receive enable (RXEN), and Transmit enable (TXEN) ports to switch between these two modes on the UART. Correspondingly, the UART receive and transmit data lines are connected on both the MCU and the transceiver. The transceiver is set to transmit at its highest power using its PDN port, but the level adjusting pin (LVLADJ) is connected to an MCU port to provide control over the power level of transmission. Also the RSSI indicating pin is made available to an analog port to take advantage of RSSI values for networking. This feature can aid in incorporating routing algorithms such as the LEACH [15]. The remaining digital ports on the microcontroller are attached to LED’s which are connected to ground through resistors, and serve as display and testing methods. The crystal is attached to the MCU’s two oscillator pins to provide the external clock, and the sensor connector’s sensor input line is connected to an analog channel on the MCU.

2.3 Smart Stone Board Layout

To minimize the cost of prototype manufacturing, the Smart Stone PCB was restricted to a 2-layer design which was manufactured using the barebones option available through Advanced Circuits [32]. Through-hole components were chosen to enable easy soldering to rapidly complete the development process.

Several considerations were taken into account while developing this 2-layer board. The bottom layer is dedicated to creating a *ground plane*. A ground plane provides ground and distributes power better than traces. It is particularly important in this circuit due to the presence of Radio Frequency (RF) electronics on the board. The ground plane prevents radio waves from antennas unintentionally formed by tracks. In addition, the presence of a ground plane eliminates ground loops which can be inadvertently formed when several ground connections on different components are connected in series, rather than to one single centralized ground point. These series connections promote miniature loops (circuits) between each individual piece of equipment, allowing RF current to circulate at differing intensities, which are another source of radiated RF noise. As the ground circuits "float" above zero potential they never actually draw down to true ground. A dangerous shock hazard to the operator can result, but is easy to avoid through good design practices. To avoid this phenomenon,

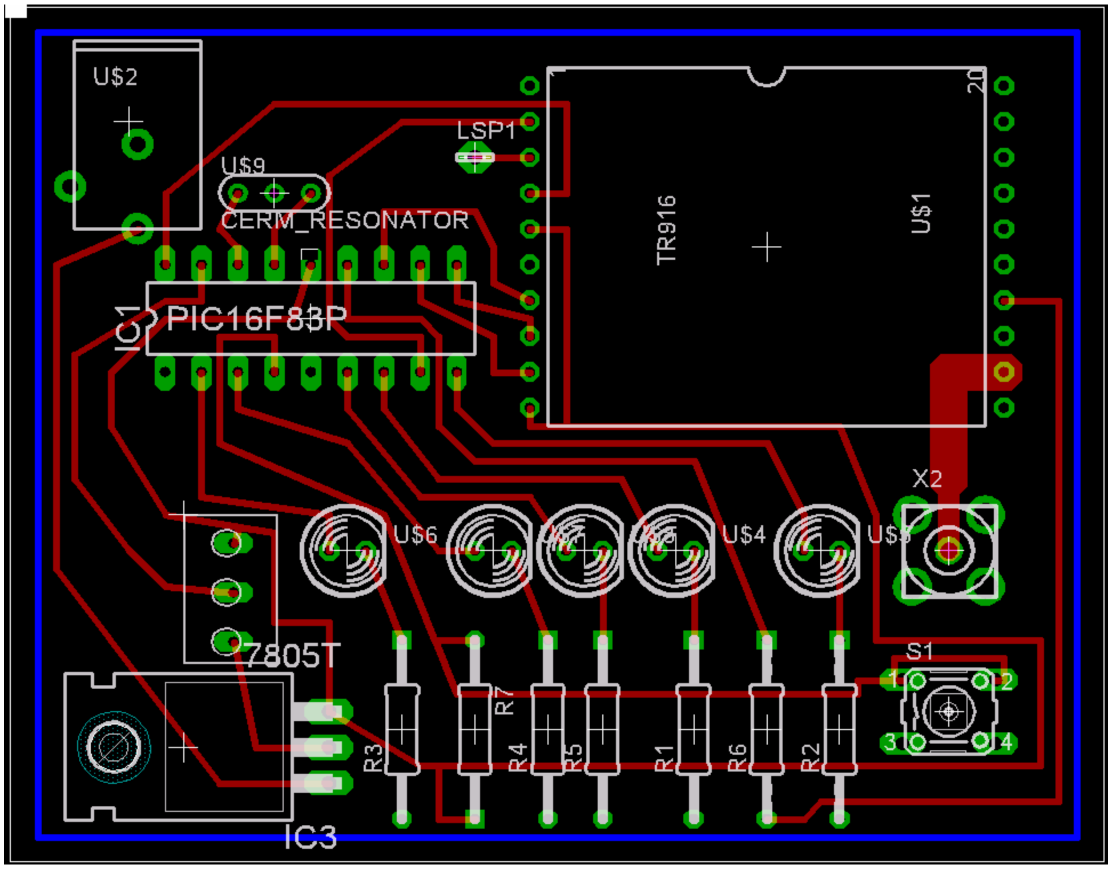

Figure 2.3 Board Layout for the Smart Stone PCB

each component ground should be directly connected to the ground plane using vias with the shortest possible traces. The ground plane is stretched in the Smart Stones over the entire board area. As shown in the board layout in figure 2.3, the transceiver is placed in an isolated corner with minimum traces underneath the RF module to prevent coupling. In particular, the TR-916-SC-P is situated away from high frequency crystal. The antenna is also placed away from all the other components. All traces supplying power are made thicker (twice the regular traces) to reduce impedance of these traces while carrying larger currents. The same is implemented for the RF signal to the antenna. An *sma* connector was chosen to attach to the antenna. Separate vias are not required to connect the component grounds to the ground planes since all parts are through-hole parts and grounds get directly tied to the plane. In case of surface mount parts vias would be necessary.

2.4 Smart Stone Mote Manufacturing

After designing the board it was subjected to an electric rule check (ERC) where the board is tested against the schematic for consistency. Further, a design rule check (DRC) was performed to ensure the design met manufacturing requirements. The DRC checks for sufficient trace widths, component and trace spacing, and other features that are required to properly manufacture the board. Insufficient spacing can cause shorts while soldering. Upon successfully passing the ERC and DRC, a CAM processor was utilized to convert the layout into Gerber files. Four files were generated with extensions *.cmp, .sol, .drl, .drd.* These files contain information regarding the positions and layers for each component, and the desired drill sizes and hole locations on the board. The *.cmp* and *.sol* files detail the parts on the top and bottom layer respectively, whereas the *.drl* and *.drd* files detail the size and positions of holes and vias on the board. These files were zipped and uploaded to the Advanced Circuits website [32] for a final check using their Free DFM feature to check the design against their specific manufacturing requirements. The final board was designed to be 3 inches X 2.35 inches.

After board manufacturing, the final considerations involved features to ensure the microcontroller could be reprogrammed and that both the microcontrollers and transceivers can be used in other applications when needed. To incorporate this, an 18 pin socket was soldered to the board which the MCU fits into and can be easily removed for reprogramming. Similarly, pin rows were soldered to fit the transceiver to facilitate removal when required. Figure 2.3 is a photograph of the Smart Stone equipped with a delocalized acoustic sensor. Figure 2.4 shows a 5-node Smart Stone network performing the synchronization algorithm which will be discussed in Chapter 3.

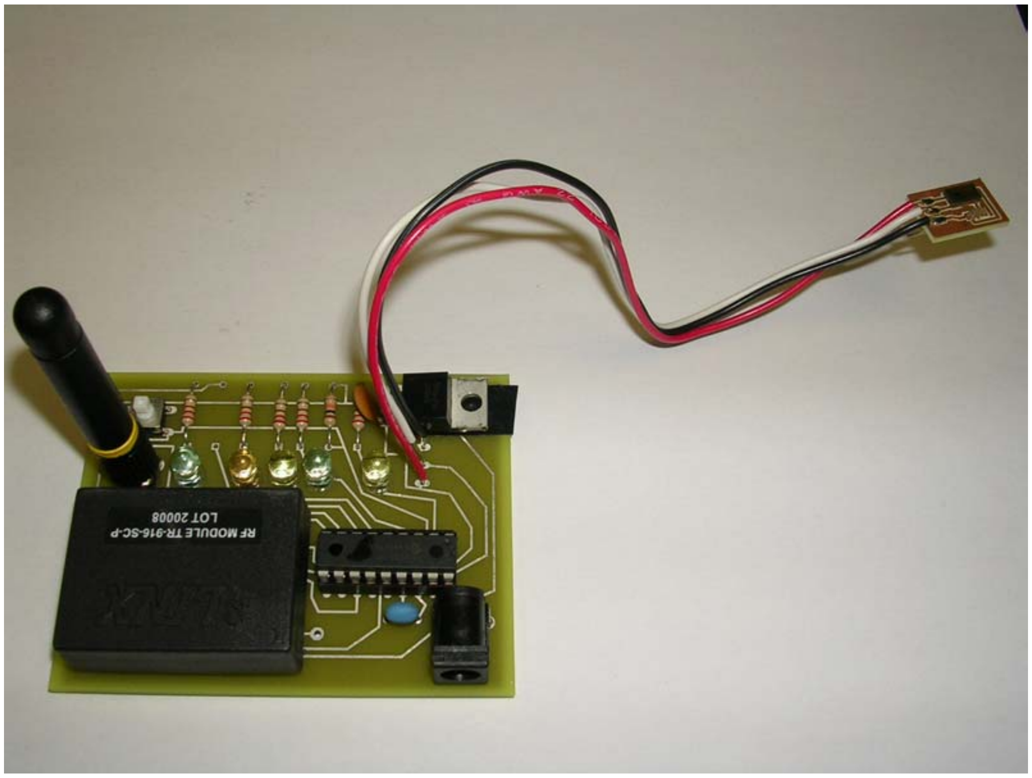

Figure 2.4 Photograph of a Smart Stone with an Acoustic Sensor

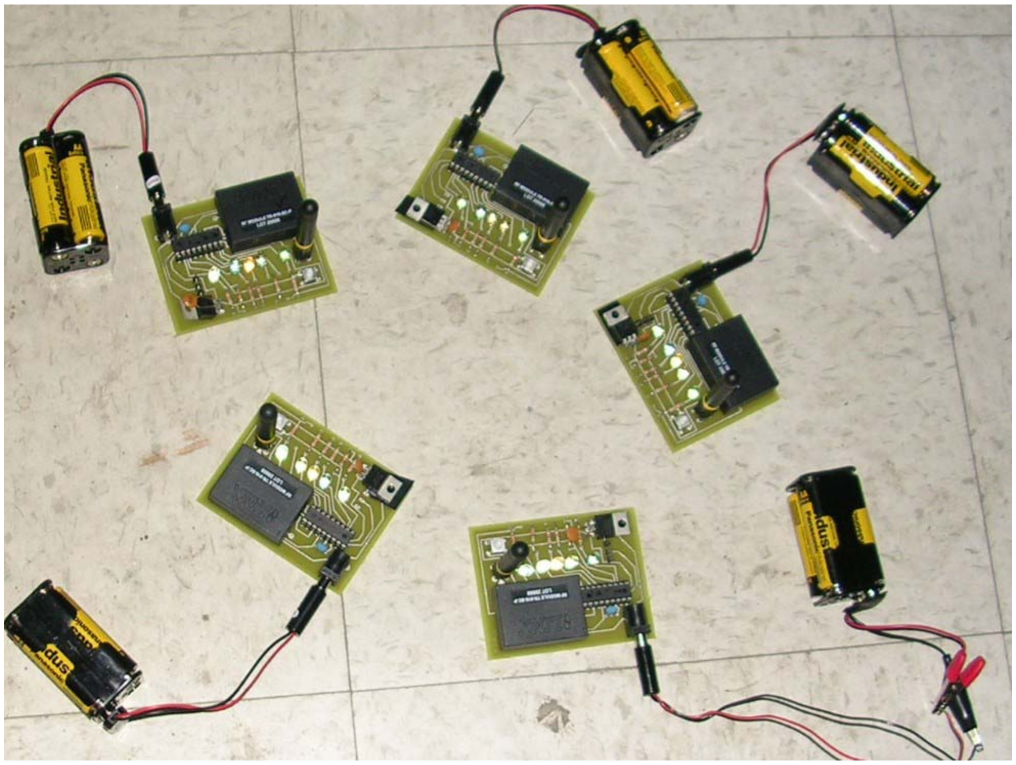

Figure 2.5 a) 5-node Smart Stone Network

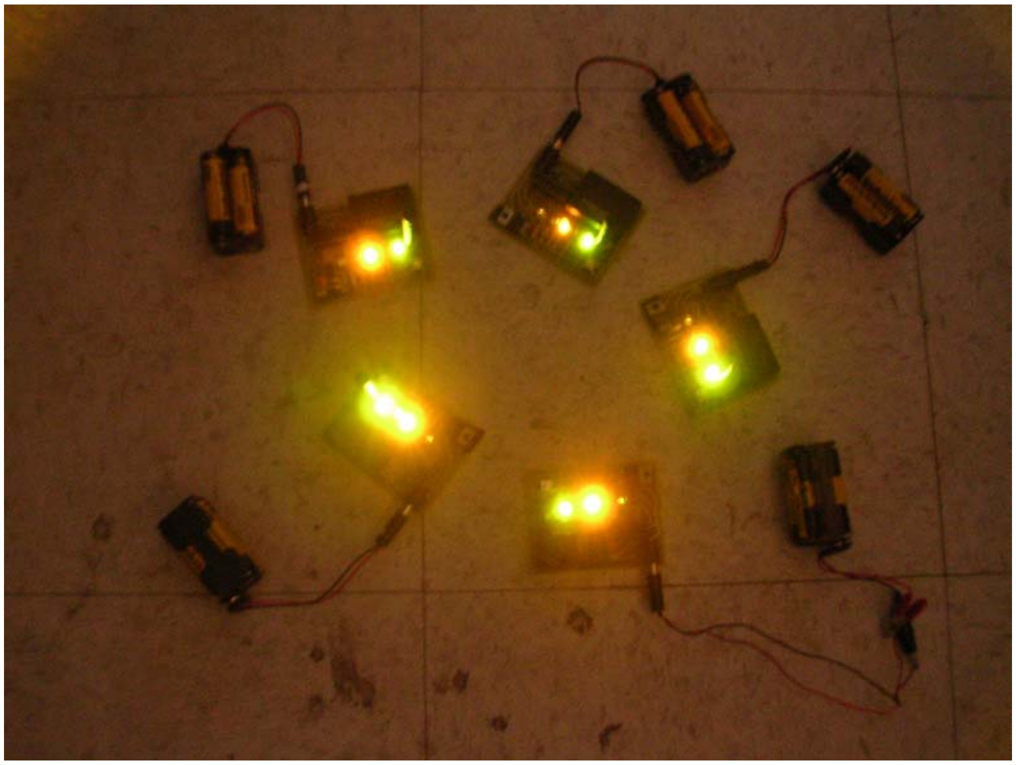

Figure 2.5 b) 5-node Smart Stone Network performing synchronization algorithm

CHAPTER 3

3. Networking and Sensing Algorithms for Smart Dust: The Smart Stone Protocol (SSP)

A thorough survey of the existing clock synchronization algorithms and MAC protocols presented in Chapter 1 revealed the efforts and progress of the research community in approaching the difficult problem of establishing low-power communication in sensor networks. The studied pros and cons of each proposed method also suggested the need for algorithms that consider as many of the limitations of WSN's as possible while achieving robust networking. In addition, conclusions were drawn that proposed algorithms should be more hardware oriented for successful operation on sensor motes, and tested on sensor platforms to validate their practicality. The literature survey presented very few protocols that had actually been tested on existing wireless sensor nodes. Besides developing smarter clock synchronization algorithms, the necessity to construct mechanisms to setup robust communication in spite of the synchronization limitations was considered to be a principal focus of the effort. In this chapter an innovative, low-power TDMA based synchronization algorithm and communication scheme is proposed. The Smart Stone Protocol (SSP) has been tested on the Smart Stones described in Chapter 2, and demonstrates rapid synchronization, and robust direct communication. The experiments and steps involved in developing the algorithm are discussed in detail in this chapter followed by a thorough analysis of the SSP.

3.1 Time Division Multiple Access Algorithms

As described in Chapter 1, sensor nodes which perform distributed computing, must communicate with each other over a wireless connection. Sensor networks have unique limitations compared to other wireless networks. Envisioned to run on battery power for months or years, power management becomes the most important issue in wireless communication for these networks. In particular, transceiver operation [7] usually is the bottleneck with regards to power. Both transmissions and receptions consume more power than any of the processing performed by the nodes [7]. Communication algorithms can be designed for sensor networks in general, however, considering specific requirements and limitations of a particular sensor network will result in the most power efficient protocol. These requirements and limitations include the density of nodes in an area, their topology, the number of nodes that need to share information, the presence of special, more powerful nodes, and other similar considerations. This introduces the need for medium access control (MAC) protocols which govern the scheme in which the network nodes gain access to the medium of communication. As discussed earlier, most sensor nodes are not equipped with transceivers transmitting at different frequencies, resulting in a single frequency band to be shared by all the nodes. Two nodes transmitting at the same time may interfere with each other at all points where their interference range is common, resulting in corrupt data. In these cases, the nodes must transmit again wasting both their own energy and the energy of the receiving nodes. This notion suggests the need for Time Division Multiple Access protocols which assign specific slots to each node to transmit to prevent collisions, since avoiding collisions is the most power saving method in WSNs.

3.1.1 Clock Synchronization based on Timers

In order to set up transmission slots and to make sure that each node transmits at the correct time, the entire network, or the section of it on which TDMA is established must have a common notion of "time." This necessitates the requirement of a common clock. The presence of *N* nodes in a TDMA setup implies the need for possibly *N* slots for the nodes to report in. These *N* slots form a single *frame*. A frame may require additional slots for discovery of new nodes typically governed by a master node or a base station, usually incorporated into the control phase of the TDMA frame. The size of the slots is dependant on the amount of data to be transmitted by each node, and also by the time taken to switch the transceiver between transmit and receive modes. This scheme requires each node to be aware of the time a frame starts with high precision, and a method to resynchronize their clocks to prevent clock drift over time.

3.1.1.1 2-node Clock Synchronization (Wired Setup)

Almost every commercially made microcontroller is equipped with hardware timers. Hardware timers comprise registers whose value is incremented or decremented in hardware upon completion of a certain period governed by the system clock and the timer configuration. An initial concept was to attain clock synchronization amongst two nodes to test the idea of using timers. To eliminate the intricacies of radio transmissions, a wired setup was created on a breadboard to simulate transmission and reception between two nodes. The UART transmit port of the microcontroller on each node was connected via a wire to the UART receive port of the other, facilitating data transfer between the nodes. This simulation proves to be very realistic given that in a wireless implementation these UART ports are connected to the transceiver transmit and receive pins.

TIMER0, an 8-bit timer in the PIC 16F88 was chosen to realize the synchronization. The *TIMER0* module increments every instruction cycle, in the absence of a pre-scalar. The *TMR0* interrupt is generated when the value in the *TMR0* register overflows from *FFh* to *00h*. The transmitting node, node *A*, was programmed in assembly language to run *TIMER0* and transmit this timer value once to the receiving node, node *B*, which was programmed to receive the timer value and assign this value to its own timer. Hardware was set up to enable both nodes to send their timer value to the computer at the press of a common button. On conducting the experiment and pushing the button, the values displayed were different and were not constant for several repetitions of the experiment. The reason for the disparity was that the time taken to send the value over the UART to the second node was of the same order as the timer increment count. In addition, the *send* and *access* times, discussed in Chapter 1, are often non-deterministic.

3.1.1.2 2-node Clock Synchronization using third node (Wired Setup)

A second experiment was conducted sending the same information from node *A* to two nodes *B* and *C*, and examining the timers of *B* and *C*. Incorporating the third node in the breadboard setup however posed certain issues. Each PIC16F88 provides a single UART with one transmit port and one receive port. Two transmit lines cannot be directly connected to a UART receive port since this results in corrupted data. The transmit line on the UART is held logic high when the UART is receiving data. To allow the correct data to enter the receive port, the transmit lines from the other nodes are passed through an *AND* gate as shown in figure 3.1 and the argument for this approach is shown in the truth table in table 3.1. Upon pushing the button, the timer values for both *B* and *C* were exactly the same over repeated executions of the experiment. The findings though encouraging, introduced the need for a master node which would be used for synchronizing the slaves but after synchronization would not be synchronized with the network itself, rendering the technique impractical for setting up TDMA slots. Analyzing the experiments conducted, however, suggested that this concept could prove effective if the time taken to transmit the information was insignificant compared to the timer intervals.

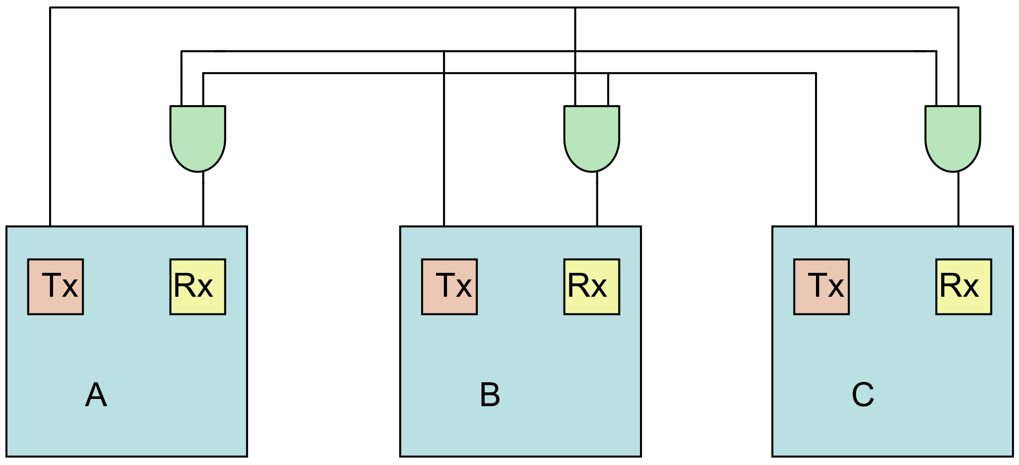

Figure 3.1 Block Diagram for 3-node wired network

Tx line (B) = P	Tx line (C) = Q	Rx Line (A) = P AND Q
0	1	0
1	1	1

Table 3.1 Truth Table for Node A receiving from Node B while Node C transmit line is held high

3.1.1.3 Clock Synchronization based on derived Timers

To realize synchronization using timer intervals larger than that allowed by the hardware timers, the notion of derived timers was introduced. A software timer, *localTime* was created using a register and programmed to increment every time the *TMR0* interrupt occurred. This slowed down our timer to 1/256th of the previous increment interval. The experiment to transmit and display the timer value was repeated for Nodes A and B, but this time transmitting *localTime.* The values displayed were exactly the same each time the experiment was conducted and the results were identical when the third node, node C was introduced. The resulting derived timer hence provided the nodes with a new sense of a clock which could be used to co-ordinate transmissions and receptions in the network.

The next step was to set up a network using this scheme of clock synchronization. Three nodes were programmed to run *TIMER0*, and increment *localTime* on the *TMRO* interrupt. Nodes A, B and C were assigned unique identification numbers, *nodeIDs*: 1, 2 and 3 respectively. The nodes kept track of their transmission turns using a register *transmitID* which was initialized to 1 in each node. In the algorithm, every time *localTime* overflows from *FFh* to *00h*, the node checks if its *nodeID* and *transmitID* are the same. If identical, the microcontroller transmits a packet comprising 3 bytes. The first byte is a frame sync byte and is used to identify the network. Reception of this byte indicates a packet being transmitted within the network, and marks the beginning of a predetermined set of bytes to be decoded by the receiving node. In the Smart Stone Network *CBh* is used as the frame sync. The second byte is the node identification number followed by the third byte which is the current *localTime*. The receiving nodes, whose *nodeIDs* do not equal *transmitID,* await the frame sync byte. Upon reception, they receive the *nodeID* of the transmitting node followed by the *localTime* of the transmitting node, and reset their *localTime* to 00h. After transmitting or receiving, *transmitID* is increased by 1 if the old *transmitID* is 1 or 2, and reset to 1 if the old *transmitID* is 3. This algorithm sets up the slot schedule shown in table 3.2.

	Slot1 *transmitID* = 1	Slot2 *transmitID* = 2	Slot3 *transmitID* = 3
NodeA	**Tx**	Rx	Rx
NodeB	Rx	**Tx**	Rx
NodeC	Rx	Rx	**Tx**

Table 3.2 Slot Schedule for 3-node network

This algorithm, outlined in figure 3.2, though effective in synchronizing the network, and power efficient in requiring only 3 bytes of transmission, has significant limitations. In particular, depending on *transmitID* to equal *nodeID,* necessitates node A to be the first to transmit. In cases where node A may be missing or out of range additional algorithms need to be implemented to ensure that one of the other nodes starts the transmit cycle. Robustness is a critical feature for the functioning and practicality of any network. A robust network should be able to synchronize itself irrespective of the absence of expected nodes, should be capable of allowing new nodes to join the network, and allow nodes to re-enter the network if they had fallen out of range. To increase the robustness of the network, the nodes were programmed to increment the *transmitID* when *localTime* overflowed, to ensure that a missing node did not stall the network. In addition, a node which was a part of the network, after not receiving transmissions for *N* slots, where *N* is the network size, was programmed to consider itself out of range of the other nodes and revert to receive mode. This was implemented because nodes desynchronize over time, and after coming back in range, the re-entering node might cause interference. Reverting to receive mode allows the re-entering node to receive the first transmission when it comes back in range, and use the *nodeID* of the transmitting node to reset *transmitID* for the next slot.

3.1.2 Wireless Communication in the Smart Stone Network

After developing algorithms using the wired connection between the nodes, wireless links were to be set up to replace the wires. As in the wired UART implementation, the wireless channels can only receive or transmit at a time. On the breadboard setup, LINX transceivers were placed, and the transmit and receive lines of the transceiver were wired to the respective UART channels of the microcontroller. Added considerations were, allowing enough startup time for the transceiver, and switching time between receive and transmit mode within the slots. The timer synchronization precision needed to be checked again, since the delays of *access* time, *propagation* time, and *receive* time increase when using the transceivers. In addition, the transmitted packet had to be modified to set up the wireless link.

Using the same algorithm as used in 3.1.1.3, on startup, the microcontroller waits for the transceiver to be ready for valid transmission or reception. For the TR-916-SC-P this could be a maximum of 8ms. For a transmission slot, the microcontroller sets *TXEN* high, and *RXEN* low, where *TXEN* is the transmit enable pin, and *RXEN*, the receive enable pin. Both cannot be high at the same time. The converse is performed for a receive slot. The packet was modified to comprise to following:

[*RF sync byte*] [*UART sync byte*] [*Frame Sync Byte*] [*NodeID*][*LocalTime*]

The RF sync byte is a code *55h* (01010101) which enables the receiving transceiver to lock onto the incoming transmission. The UART sync byte *FFh*, ensures that the start bit of the frame sync byte will be accurately detected. The UART interprets the start bit of a byte as a 1-0 transition which makes it difficult to detect the start bit if *55h* (01010101) is received. The UART sync byte *FFh* (11111111) creates a high marking period of one byte so that the start bit of the following frame sync byte is accurately detected. The frame sync byte, as discussed earlier, is then used by the microcontroller to intelligently identify the start of the packet, and interpret the following bytes in a pre-determined order. After transmitting, the nodes switch to receive mode which takes up to 6ms, and the node whose *nodeID* equals *transmitID* switches from receive to transmit mode which takes up to 5ms.

After thorough breadboard testing, the algorithm, implemented in assembly language, was rewritten for the Smart Stone nodes taking onboard connections into consideration, and downloaded into 3 Smart Stones. The Smart Stones demonstrated the same performance as the breadboard setup, achieving synchronization and maintaining this over time to function as a network. While extending the algorithm to more than 3 nodes, important limitations of the algorithm were revealed. Even though the algorithm is well-scalable, the smallest slot size achievable was approximately 200ms with a pre-scalar of 1:16 on *TIMER0*. The technique of synchronizing the nodes by equating timers is inaccurate once the pre-scalar is further reduced. Therefore, to accommodate 8 nodes, the entire frame length would be 1.6s with 200ms slots, and for 128 nodes, each node would report every 20s which maybe considered impractical for some sensor networks. However, each node uses less than 1/10th of 200ms to transmit its information, resulting in a significant waste of time, and receiving energy in the network. These issues coupled with the desire for more robustness, gave rise to the final algorithm used in the Smart Stone Network described in section 3.2.

3.2 Innovative TDMA-based algorithm used in the Smart Stone Network

The final algorithm implemented in the Smart Stone motes was designed to acquire rapid synchronization amongst all nodes, demonstrate robustness, satisfy essential requirements of sensor networks, and also eliminate some of the known shortcomings of the previous method being used. TDMA, though a highly lucrative method of communication given the low occurrence of collision interference once synchronization is achieved, involves the transmission of extra synchronization bytes to make the network aware of the global clock. These added bytes increase the power required to transmit the entire packet serving as a drawback of the TDMA approach. The algorithm proposed meets the following stringent requirements

1. Attains rapid synchronization amongst nodes
2. Synchronization remains over long periods of time
3. No master node required, nodes can be turned on in any sequence
4. Nodes can exit and re-enter the network without interfering or stalling the network
5. No extra synchronization bytes are sent to acquire the synchronization, hence extremely power efficient
6. Large number of nodes can be incorporated while reporting within a practical frame length

The algorithm while taking all above requirements into consideration, also involves additional steps to demonstrate the synchronization to an observer, all of which can be removed in a final problem oriented version. The communication algorithm of the Smart Stone protocol (SSP) is shown in the flowchart in figure 3.3.

3.2.1 Initialization of Algorithm

The first step is an initialization procedure. Each node is assigned an identification number called *nodeID*. This assignment is the only difference in the code downloaded into each node. The rest of the code is identical on all nodes. This proves to be an important feature for distributing and commercializing the code, and for downloading the code onto the microcontrollers. Next, registers, memory locations, and ports are set up to realize the algorithm. The baud rate for the UART is initialized in a special register *SPBRG* to transmit and receive at 19.2Kbps. The initialization is dependent on the system clock which is 20Mhz in our motes. *TIMER0* is programmed to run with a pre-scalar of 1:256 which implies that the timer register value increments every 256 instruction cycles, or 50µs. A delay function is invoked to enable the transceiver to start up after which the *TXEN* pin is pulled low, and the *RXEN* pin is pulled high to set all nodes in receive mode. With the device ready to perform transmissions and receptions, the global interrupt is turned on to allow the *TMRO0* interrupt to flag a timer overflow. Finally, a green led is turned on to show the observer that the device is powered on. A red light is also turned on when the node is waiting in receive mode.

3.2.2 Main Receive Loop

Upon initialization, the program enters a receive loop titled *_syncwait.* In the loop, the microcontroller waits for a received byte. Once a byte is received, a check is performed to see if the byte equals *CBh*, the frame sync byte. Any other byte might be transmissions alien to the Smart Stone network, or radio or UART synchronization bytes, which should be ignored until the frame sync is received. If the byte is not equal to the frame sync, the program returns to the *_syncwait* loop. If equal, the microcontroller waits fort the next byte which is the *nodeID*. Reception of this byte requires elaboration as the synchronization algorithm is intelligently based around this byte. The packet being transmitted by the transmitting node has been shortened to be

[*RF sync byte*] [*UART sync byte*] [*Frame Sync Byte*] [*NodeID*]

The *localTime* from the previous algorithm is not separately transmitted, even though it may be non-zero, since the timing information is contained in the *nodeID*. Acknowledging the drawbacks of the large slot size in the previous algorithm, and the need to eliminate additional clock synchronization bytes in TDMA packets, a scheme has been developed to distribute *nodeID's* within the 256 counts of *localTime*. Instead of transmitting when the *localTime* reaches 00h, nodes transmit when *localTime* equals their *nodeID*. The *nodeID* transmitted, is the *localTime* at the transmitting node, and all receiving nodes must synchronize themselves to this value of *nodeID*. This not only reduces a byte of transmission but also allows us to test the limits of decreasing the slot size by assigning *nodeID's* separated by smaller intervals. The limits of this interval are discussed in section 3.2.4. The receiving node, thus, receives the byte after the frame sync byte as the *nodeID* of the transmitting node, which is also the *localTime* at the transmitting node, and sets its own *localTime* equal this received time. *TIMER0* is reset at this time too. To aid the observer, a green LED is turned on every time the frame sync is received and turned off after synchronizing, which results in a flicker of the green LED when a valid network reception has been made. After receiving and decoding the packet, the node returns to the *_syncwait* receive loop.

3.2.3 Interrupt Triggered Transmissions

The only interrupt used in the algorithm is the *TMR0* interrupt which indicates the *TIMER0* value overflowing from *FFh* to *00h*. The Interrupt Service Routine (ISR) is programmed to disregard all other interrupts, if triggered, and return to the main program. If the *TMR0* interrupt occurs, the *localTime* is incremented, and then tested against *nodeID*. If equal, it is the node's turn to transmit, otherwise the receive loop is executed. In case of a transmit turn, the node switches from receive to transmit mode by pulling *RXEN* low and *TXEN* high. A yellow LED is turned on to indicate transmit mode. The node transmits the packet, one byte at a time, starting with the RF sync byte *55h*, followed by UART sync byte *FFh*, the frame sync byte *CBh*, and its own *nodeID*. The microcontroller then ensures that the transmit buffer is empty and the transmission is complete, and loads *nodeID* to *localTime*, to make the time at which the timers are reassigned values, as similar as possible on the transmitting and receiving nodes. *TIMER0* is reset again, and the microcontroller then switches back to receive mode and waits in the receive loop. This last step is a crucial part of the algorithm and differentiates the algorithm from other clock synchronization schemes that result in the transmitting node remaining desynchronized from the receiving nodes [16, 18].

3.2.4 Assignment of *nodeIDs* for synchronization

Each node enters the network in receive mode, ready to transmit when *localTime* reaches *nodeID.* This ensures that each node is synchronized to the rest of the network immediately after receiving the first transmission. The time at which the transmit buffer gets empty on the transmitting node, and the time the receiving node receives the *nodeID*, are the closest comparable times on these separate devices, which justifies our decision to reload *nodeID* into *localTime* on the transmitting node, and load the received *nodeID* into *localTime* on the receiving nodes. The slight difference that may occur is insignificant and is not an error that builds up or causes the clocks to drift apart. Our algorithm makes the entire network resynchronize itself on every reception with no additional data transmission or power consumption. The error, therefore, never accumulates but is rather constant over time. In addition, our experiments have revealed that given the identical hardware and software on the nodes, the transmitting node may have a marginal difference in its clock from the receiving nodes, whereas the receiving nodes share the same clock value and are truly synchronized. Besides accurate clock synchronization, the algorithm also removes the need of a master node to schedule the slots in the network, allowing the network to synchronize without dependence on any particular node. Master nodes may still be used for clustering applications, but are not responsible for cluster synchronization.

In the previous algorithm, the smallest slot size possible was 200ms. We concluded that this was impractical for larger networks. Using the final algorithm, the default frame length has been set to 3.35s. All nodes report within this time. *NodeIDs* can be assigned depending on the number of nodes in the network. For example, an 8 node network may be assigned *nodeID's*: *10h, 30h, 50h, 70h, 90h, B0h, D0h, F0h,* and each node transmits when its *localTime* reaches the *nodeID* value. Equally spacing the *nodeIDs* ensures that the slots are balanced within the frame length. Experiments were conducted to test the maximum number nodes that could be accommodated within the 3.3s frame without interference. To achieve the goal, three nodes were programmed starting with *nodeIDs* 4 counts apart (*C0h, C4h, C8h*). The node with *nodeID C8h,* after transmitting, also sent a list of all the *nodeIDs* it had received transmissions from, to the computer. The experiments showed perfect receptions for this case, and also when the *nodeIDs* were 2 counts apart (*C0h, C2h, C4h*). However, using consecutive *nodeIDs* (*C0h, C1h, C2h*) caused interference and the packets were often not received by each node. This demonstrated that the tasks to be performed for each transmission/reception slot took longer than 13ms. Hence, the smallest slot achievable using the default settings is 26ms. The conclusion made was that using the default setup, the Smart Stone network could accommodate 128 nodes spaced 2 *nodeID's* apart from each other. The 5ms time, required to switch from receive to transmit mode, takes a significant percentage (40%) of the slot, and by switching to transmit mode just before the transmit slot we might be able to accommodate more nodes. If fewer nodes were used and 3.3s was deemed to be too long a frame length, the 3.3s can be reduced to a 200ms frame length. This, however, reduces the number of nodes that can be accommodated in a single frame. In addition, if much larger networks are to be constructed, possible ways of using the same concepts would be clustering, with each cluster using our synchronization scheme, or resorting to 2-byte timers such as *TIMER1*, which could allow thousands of nodes to be part of the network.

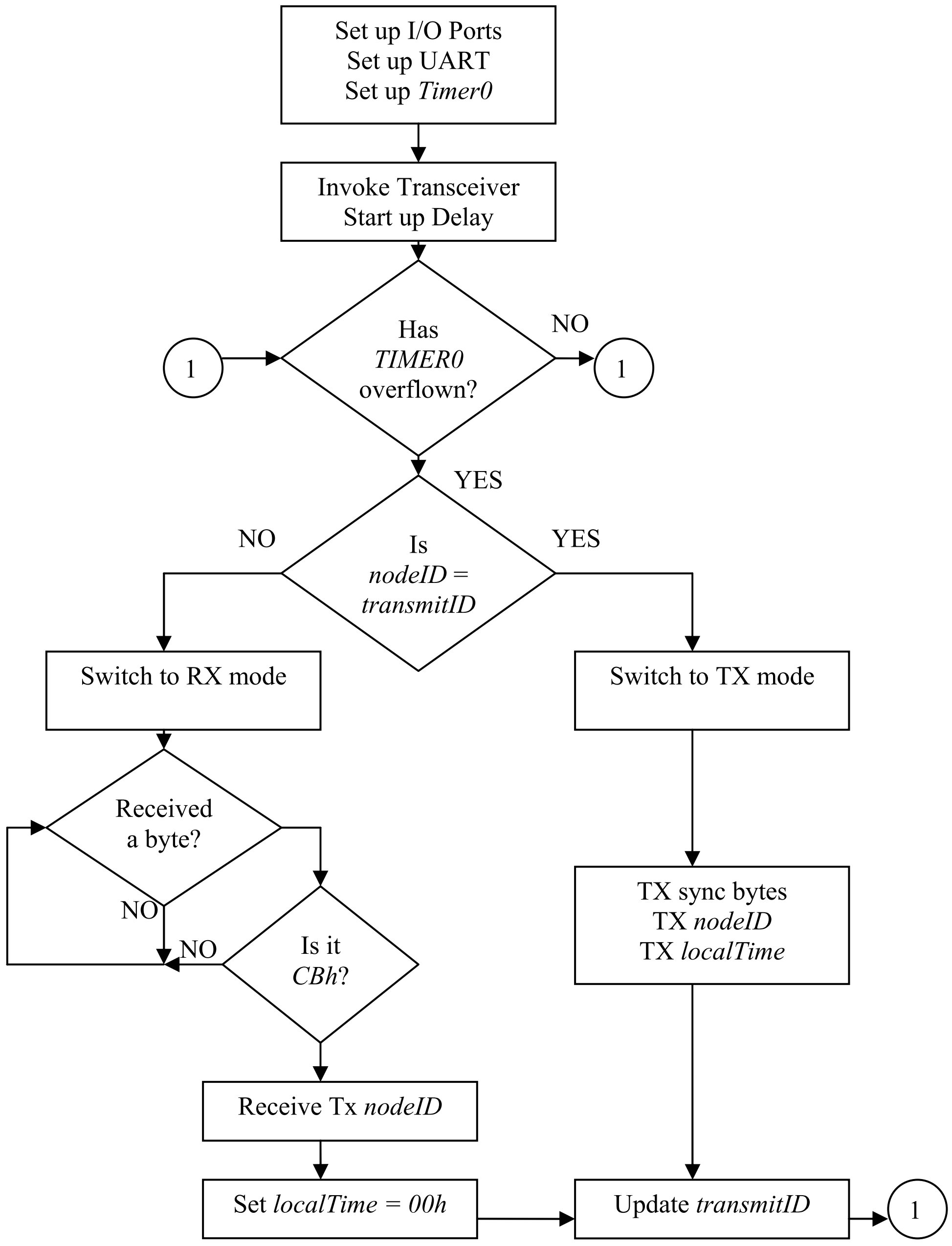

Figure 3.2 Flowchart for Algorithm discussed in Section 3.1

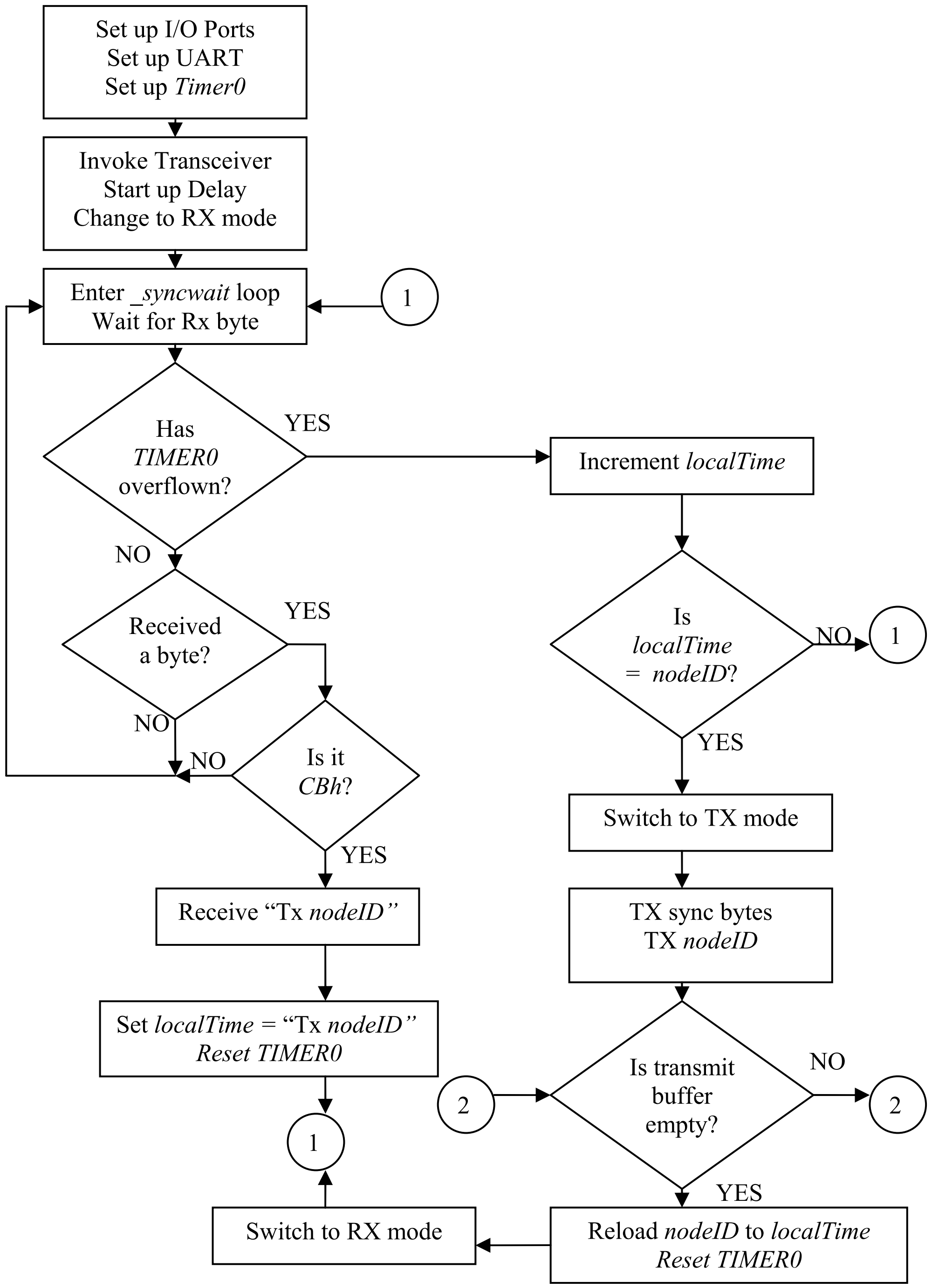

Figure 3.3 Flowchart for the Smart Stone Protocol (SSP) discussed in Section 3.2

3.3 Smart Stone Sensor Network

The primary goal of the Smart Dust project is to distribute tiny nodes equipped with sensors to perform a specific task by sharing information with the other nodes in the network. Achieving synchronization is just the first step in this process, enabling the nodes to share vital information reliably using the least amount of power. Our TDMA based algorithms ensure that nodes are synchronized and that each node can transmit information without interfering with each other, reducing the need to re-transmit their packets. Our wired setup was used as a platform to develop algorithms for sensor information processing and sharing.

3.3.1 Wired Network with Potentiometers as Sensors

The breadboard 3-node wired network was modified attaching potentiometers to each node to provide the sensor input. The problem was defined to synchronize the nodes using our technique described in section 3.2, read voltages on each node from the potentiometers in real-time, transmit the sensor readings in the packet, and have each node light up an LED if its own voltage was the highest. Almost all sensors for monitoring temperature, acoustic vibration, etc, output an analog voltage representing the intensity of the measured signal. Our problem statement encompasses the challenges faced in using most sensors, and also the task of locating the node with the most critical reading.

The PIC 16F88 is powered with 7 onchip A/D converter channels which are capable of inputting analog data. An A/D conversion results in a 10-bit digital representation of the input signal with respect to the A/D reference voltages. This 10-bit result after conversion is contained in two single byte registers. The potentiometers were connected to one such port with the supply voltage of 5V and ground as the reference voltages for the conversion. Reading the sensor input was incorporated into the transmit slot, and comparisons of the sensor readings of the three nodes was incorporated into the receive slot. The initialization phase of the program was modified to include the A/D module setup, choosing *Channel 0* to input the sensor reading, and perform the conversion at the maximum rate of *Fosc/32*, where *Fosc* is the oscillator frequency. The 10-bit result was programmed to be right-justified with the six most significant bits of the high byte containing zeros. The module was turned on after the initialization to perform conversions. In the timer interrupt driven transmit slot, when *localTime* equaled *nodeID*, the microcontroller started off an A/D conversion by setting the A/D *GO/DONE* bit high, and polled for the completion of the conversion. Once complete, the two result bytes were loaded into *A2Dhighbyte*, and *A2Dlowbyte* respectively. The transmitted packet for the wired setup was modified to be:

[*Frame Sync Byte*] [*NodeID*][*A2Dhighbyte*][*A2Dlowbyte*]

The transmission, therefore, included the sensor information gathered just before transmitting. On the receive loop, the frame sync was again used to determine the start of a valid packet, *nodeID* to resynchronize the nodes, and the incoming high and low bytes of the sensor reading were stored in particular registers based on the *nodeID* of the transmitting node. After receiving the entire packet, the microcontroller compared the voltages received from the other two nodes with its own reading, to determine the node with the highest sensor voltage. Thus, all nodes were aware of the node with the highest reading at all times. The node also turned on an LED if it deemed its own reading to be the highest. The setup was tested with a frame length of 3.3s to be able to observe the changes, and the nodes successfully lit up their LED on being supplied the highest network voltage. The network continued performing correctly while the potentiometer values were changed in real-time.

3.3.2 Smart Stone Sensor Network with *MEMS*-Microphones

The *Line in the Sand* problem, one of the proposed Smart Dust applications, necessitates the use of microphones to sense acoustic vibrations to detect intrusion across the line. Acoustic *MEMS*-microphones were acquired to be tested for use in the network. The signal was studied, and revealed a dc offset of approximately 1V. First, a PIC16F88 was used to control the microphone on a small breadboard. A/D conversions were continuously performed, and the resulting 10-bit value was left-justified this time, to store the 8 MSBs of the reading in the high byte. The two LSBs of conversion were left in the low byte and ignored for ease. Every resulting value was tested against a threshold, and if found to be greater, an LED was turned on. The threshold was tuned until it represented a value above which a footstep could be interpreted from the sensor reading.

This procedure for analyzing the *MEM's* microphones was inserted into the Smart Stone Sensor Network in conjunction with the synchronization algorithm described in section 3.2. In accordance with a line crossing algorithm being developed, the sensor nodes were to communicate to each other whether they sensed a sound above a predetermined footstep threshold. The transmit slot was edited to start off an A/D conversion on the microphone, test the most significant 8-bits (A/D high byte) against the threshold, set a footstep detection flag high to note the footstep occurrence, and send this information in the transmission packet which had the following format:

[*RF sync byte*] [*UART sync byte*] [*Frame Sync Byte*] [*NodeID*][*Threshold Flag*]

where *Threshold Flag* is 1 if a sound as loud as a footstep is detected, and 0 if not detected. During the receive slot, the receiving nodes recorded whether the *Threshold Flag* was high for the transmitting node and saved this information for the frame. LEDs were assigned on each node for the other nodes and for itself, and lit up every frame if the corresponding node had detected the footstep. After downloading the code, the nodes were placed on the ground approximately a meter apart, and a test subject walked in between two of them at varying locations. All 3 nodes showed consistent lighting of LED's demonstrating the network functioning and sharing information successfully. However, when the person walked by very fast, instances were noted where certain nodes should have detected the footsteps, but failed to do so. This was attributed to a sensor reading being read only once in a frame. Walking across the line formed by the motes fast could result in certain motes not recording the incident due to insufficient sampling. The experiment brought attention to the fact that the A/D module should continuously read the microphone readings in parallel with the transmission and reception slots.

The A/D conversion module in the 16F88 is independent of the UART, and hence, can function in parallel to the rest of the functions performed in the microcontroller. In the final Smart Stone Protocol (SSP), the same initialization is performed as in section 3.3.1. The A/D conversion, instead of running sequentially in the transmit slot, is programmed to execute in parallel with the rest of the tasks. The module is set up to trigger an interrupt each time a new conversion had been made. The A/D interrupt is enabled in the start of the program, and conversion started off before entering the *_syncwait* receive loop. The A/D conversion clock is derived off the system clock. *Tad*, the time taken to convert a single bit has to be chosen between 1.6μs and 6.4μs. A 10-bit conversion takes up to 9 *Tad* plus a 2 *Tad* waiting period after every conversion. With a conversion clock of *Fosc/32,* a sensor reading is made approximately every 70μs. Upon completion of the conversion the A/D interrupt causes the program to enter the Interrupt Service Routine (ISR*)*. The ISR checks to see if the interrupt is a timer interrupt or an A/D interrupt. For the A/D interrupt, threshold checks are performed each time and if any of the sensor readings is above the footstep threshold, the *Threshold Flag* is set high. The reception and transmission slots continue exactly as the previous algorithm. The *Threshold Flag* is set to zero after transmitting to ensure that the flag is separately tested for each frame. The tests conducted using the programmed Smart Stone Network showed detection of footsteps by the nodes close to the point of crossing even on fast movement, and consistency in the shared awareness of the network. Decreasing the time between acoustic samples to approximately 70μs, ensured that the network never missed vibrations irrespective of the duration of the disturbance.

3.4 Analysis of Smart Stone Protocol (SSP)

The clock synchronization and medium access protocols were designed with due attention to the limitations in wireless sensor networks and the resulting clock synchronization metrics. The Smart Stone Protocol (SSP) was also developed recognizing the shortcomings of existing schemes. In this section, the SSP is analyzed with respect to the requirements of WSNs, and contrasted and compared with the literature review in Chapter 1.

3.4.1 Analysis of the Smart Stone Protocol with WSN limitations

The Smart Stone synchronization and communication protocols were designed while considering each of the WSN limitations discussed in Chapter 1. These limitations are revisited in this section in the context of our implemented algorithm.

3.4.1.1 Low Power

The primary limitation of WSN's is the low power budget on each node. The smart stone network synchronization protocol is designed to operate under the most stringent power limitations. The Smart Stones acquire synchronization without any additional synchronization phase. Data transmissions are necessary for sharing events with the other network nodes, and synchronization is achieved using the same messages used for transmitting data. The resulting synchronization is tight with respect to the sender and all the receivers, allowing the nodes to be put in "sleep" mode if they do not need to hear each TDMA slot transmission.

3.4.1.2 Low Bandwidth

A constant transmission rate of 19.2Kbps is currently used in the network. The synchronization scheme is devised to work around the difficulties in synchronizing the sender to the receiver, removing the need to transmit at higher rates. In addition, the intelligently devised Smart Stone protocol packet reduces the transmission burden on the network.

3.4.1.3 Small Transmission Length

Small transmission length is a simple but powerful concept in Wireless Sensor Network transmissions. The power saved in eliminating the need for one extra byte has proven to be significant in a single transmission [7]. However, the power saved is much more evident considering that each node may transmit once a second, or 86,400 times a day. For networks which are constructed to run for years, saving 1 byte of transmission in a 100 node network may save up to 8,640,000 bytes of transmission a day which amounts to a significant power advantage to the network. The Smart Stone Protocol low power features are highly dependent on the structure of the transmission packet. By transmitting at times where *nodeID* equals *localTime* we reduce a byte of transmission, and achieve slot synchronization without using a single extra synchronization byte. Given this feature, and the absence of any separate synchronization phase, the Smart Stone network synchronizes itself without using any additional power.

3.4.1.4 Limited Network Connectivity

The Smart Stones utilize constant synchronization to ensure tight TDMA slots. Every time a valid reception is made, all listening nodes are synchronized to each other and the sending node. This ensures that the nodes remain synchronized even of they miss transmissions from certain nodes due to limited connectivity. If a node falls out of the network for some time it attains synchronization on the first broadcast received making the network robust.

3.4.1.5 Large Scale Network

The Smart Stone Protocol in its current form is best suited for achieving synchronization and sensor data transmission within peer-to-peer networks, or clusters in larger networks. The scheme has been tested for immediate use for up to 128 nodes. Schemes to extend the concepts to larger networks will be discussed in chapter 4 for future work. These additions to the protocol will cover incorporation of more than 128 nodes into peer-to-peer clusters and synchronization and message passing between unconnected nodes in the network.

3.4.2 Analysis of the Stone Protocol with existing Synchronization protocols

Several issues have been highlighted and addressed in literature while proposing the existing protocols discussed in Chapter 1. The proposed schemes also introduce numerous other issues after weighing their respective advantages and limitations. The literature survey in Chapter 1 was critical to the development of the Smart Stone Protocol. This section is dedicated to an in depth comparison of our scheme with these protocols with respect to the metrics discussed for clock synchronization analysis. The results and comparisons are also summarized in table 3.3.

3.4.2.1 Synchronization Precision

Synchronization precision is a requirement of the network which varies with the MAC protocol used and the application. Our implemented scheme uses TDMA communication slots and requires synchronization tight enough to maintain these slots without interference. Our clock synchronization scheme, based on derived timers, takes advantage of the broadcast property of the medium similar to the RBS protocol [16] and the Continuous Clock Synchronization Protocol [18]. This property takes advantage of the transmission reaching the receiver nodes at approximately the same time. From the experiments reported in 3.1.1.2 and 3.1.2, we conclude that the receiving nodes are synchronized with a precision of 50µs, a single tick of *TIMER0.* This number compares extremely well to the method of Mock et al. [18] which is most similar to ours and reports a synchronization precision of 150µs. The RBS protocol [18] takes the broadcast property a step further by requiring receivers to exchange messages which contain information regarding the time they received a message from a given sender. Using these received times the nodes maintain offsets for each node in the network and report a precision of approximately 11µs. Though the precision is excellent, their method requires multiple messages sent from the sender to estimate the time of arrival at the receivers, and subsequent transmissions between the receivers to achieve this precision. Our protocol is tailored to achieve slot allocation by acknowledging the achieved 50µs precision, and does not require the level of precision offered by the RBS protocol. In particular, the power lost in the additional transmissions required to achieve the precision is not justified in the Smart Stone Protocol. In addition, the RBS protocol [16] does not correct the clock, but maintain tables of offsets for each node in the network. This is not suitable to a TDMA approach where transmission times are triggered by the clock value, thus meriting clock correction.

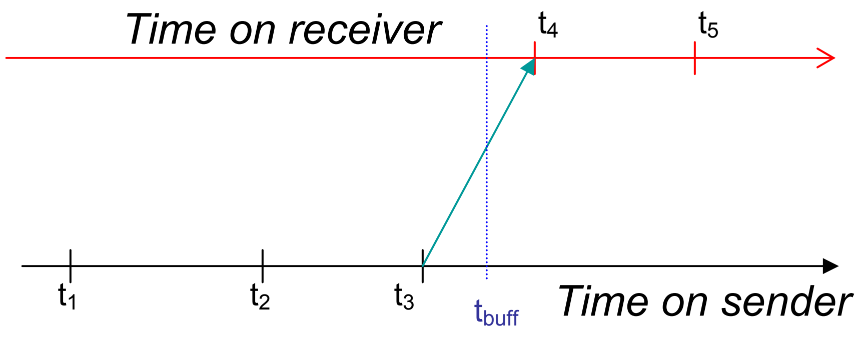

Figure 3.4 Novel Synchronization point for sender-to-receiver synchronization

Another major limitation of the RBS protocol is the sending node remaining desynchronized from the receivers. This is not acceptable in a TDMA network since the sending node might fail to enter receive mode if it is desynchronized from the rest of the network. Figure 3.4 shows real times on the sender and receiver, where the sender takes a time-stamp at t_1, starts sending the prepared message over the UART at t_2, the message is sent out via the radio at t_3, received at the receiver channel at t_4, and the correction is made at t_5. The source of the error lies in the inability to accurately determine the time taken for the sending node to prepare the report T_{prep} ($t_2 - t_1$) or the *send* time, and the time taken to transmit the report T_{tx} $(t_4 - t_2)$. The *propagation* time $t_4 - t_3$ is of the order of nanoseconds since electromagnetic waves propagate through air approximately at the speed of light. The *receive* error T_{rec} ($t_5 - t_4$), which is the time taken for the receiver to receive the message from its own channel, is also present in receiver to receiver synchronization. Traditionally, the entire time $t_5 - t_1$ is considered to be the sender to receiver delay error and may be larger than 1000µs. While attempting to reduce this error, we have identified a novel hardware time on the sender that maybe recorded, and used to achieve high sender to receiver synchronization precision. The time at which the transmit buffer on the sending node empties, say T_{buff}, is the time at which the microcontroller has sent the last byte over the UART to the radio. The UART operates using a shift register which shifts bits out one after the other to the radio. When the shifting mechanism is complete a pin *TRMT* goes high to indicate the transmit buffer is empty. The TR-916-SC-P radio does not have a buffer, and continuously transmits the bits it receives from the UART. After receiving the last byte of the message, the receiver corrects its own *localTime* to that received from the sender at time t_5 and *TIMER0* is reset. The times T_{buff} and t_5 are the closest identifiable times on the sender and receiver which are almost equal. This results from the fact that the time taken to prepare the report T_{prep} and the time taken to send the report over the UART are not a part of the resulting difference, T_{buff} - t_5. In our implementation, since the radio does not have a buffer, T_{buff} lies between t_3 and t_4, and results in high synchronization precision. Using this analysis, the *nodeID* is reloaded into *localTime* when the transmit buffer empties on the transmitting node, by polling the bit *TRMT* in the UART transmit status and control register *TXSTA*. The clock correction adjusts the clocks to within 13ms of each other, and resetting *TIMER0* at these close times provides the synchronization at the µs level. This results in the sender and receiver being synchronized within 100µs. The rest of the algorithm ensures that transmissions are made after a delay larger than the maximum skew in the network to allow each node to be in the desired transceiver mode. This is achieved by padding the transceiver switching delay. In some radios which perform error correction, the message may be stored in a buffer before being transmitted. In such a case t_{buff} would lie in between t_2 and t_3, but would still significantly reduce the delay error. In addition, these complex transceivers might be equipped with a time t_{tx_buff}, the time at which the transmit buffer on the radio empties, which would greatly improve the accuracy of the resulting synchronization.

The synchronization precisions, especially the 50μs receiver to receiver precision, are highly dependant on the baud rate used to transmit the message. At 19200 bps, the bit time, the time taken to receive one bit of information is *1/19.2k*, or 52μs. The resulting 50μs precision arises from the different times within this 52μs window that the different receivers physically receive the first bit of the message. Once the first bit is received, the other bits are received at fixed time intervals pre-decided on each microcontroller in the UART configuration. In the absence of a synchronous mechanism, these fixed times allow the asynchronous operation of the UART. Consequently, much higher precision can be achieved using higher baud rates and this is exhibited by the results presented in [16]. Finally, as pointed out in Chapter 1, Mock et al [18] might have underestimated their precision due to the lack of testing on physical platforms. However, their method requires the master to receive its own transmission, which is not possible in sensor nodes with transceivers which can only receive or transmit at a time.

3.4.2.2 Piggybacking

Piggybacking, as discussed in 1.4.1.2, is a term used to refer to the adding of synchronization acknowledgement messages to data messages sent to nodes. We extend the definition of piggybacking to include the adding of synchronization messages to data transmissions. Romer's protocol [17] and Ganeriwal et al's scheme [20] incorporate piggybacking, saving the network power while achieving synchronization. Protocols with better precision than the Smart Stone Protocol, utilize series of synchronization messages transmitted without piggybacking, draining the network's power in an attempt to achieve tighter synchronization. Given that the network's main purpose is message sharing, additional synchronization messages are essentially overheads to achieve the goal. The Smart Stone Protocol incorporates the benefits of piggybacking in the most efficient scheme possible and this proves to be one of its most lucrative features. In fact, the absence of any additional synchronization bytes in our protocol renders the scheme superior to all existing schemes in the area of synchronization overhead.

3.4.2.3 Synchronization Message Length

The diffusion based synchronization protocols [19, 22], rely on large number of synchronization messages comprising timestamps, deviations, and number of hops, propagating through the network. In contrast, the Smart Stone protocol as discussed above, achieves synchronization while piggybacking and without sending a single synchronization byte. The need for additional synchronization bytes may arise while extending the algorithm to support inter-cluster communication and applications which require additional information regarding event timestamps. These will be incorporated into the data transmission packet satisfying the piggybacking criterion. It must be noted that the more complex synchronization protocols [19, 22] extend the synchronization to the entire network, and their complexity should serve as a good benchmark while extending the SSP to be a network-wide synchronization protocol.

3.4.2.4 Convergence Time

Convergence time, the time required by the network nodes to attain synchronization is an important metric to evaluate clock synchronization algorithms. The convergence time is not just a factor at network start up. It is a recurring issue, since it is brought into play every time a new node enters the network, or nodes fall out of the network and re-enter after some time. It gains even more importance in dynamic networks where nodes change positions over time. A single desynchronized node can cause interference and disrupt the smooth functioning of the network, especially in TDMA based MAC protocols. The convergence time for protocols aimed at multi-hop networks [19, 22] are comparatively higher since they attempt to synchronize the entire network. The SSP ensures that a node is synchronized the instant it makes its first reception. In a case where all nodes are present at start up, the convergence time could be as low as 13ms. For a case where there are 128 nodes in the cluster, a new node, or re-entering node would be synchronized within 26ms of entering the network range.

3.4.2.5 Complexity

The limited power resources on each node impose a constraint on the complexity of algorithms that can be implemented on wireless sensor nodes. Complex synchronization algorithms coupled with transmission burdens diminish the mote's ability to perform signal processing tasks. The diffusion based synchronization protocols [19, 22], involve high level of complexity, but offer network wide synchronization. Mock et al's protocol [18] is the most similar to our approach. However, this continuous clock synchronization method employs the complex method of applying the clock correction continuously over a period of time. This is performed to ensure that the node does not miss time triggered deadlines. Their concern, though worthy of addressing, may not merit the increased complexity. Our protocol does not presently rely on time triggered deadlines. In case of these deadlines, we propose checking the correction interval for event deadlines, and rescheduling these events to a time in the future of the corrected clock.

3.4.2.6 Network size and Scalability

The Smart Stone Protocol has been tested on the physical Smart Stone network to work for up to 128 nodes. Though schemes have been reported to work on larger number of nodes [20, 22], these results are based on simulations and not on sensor platform implementations. Realization on physical nodes often poses issues unforeseen in simulations. The Smart Stone Protocol is envisioned to be a means of communication within peer-to-peer networks, or within clusters in networks with short transmission range. Methods to extend this protocol across the larger networks are currently under investigation. The lack of scalability solutions is the current drawback of the proposed method. However, the scheme should be easily scalable using routing algorithms such as the LEACH [15] and will be briefly discussed in the future work section in Chapter 4.

3.4.2.7 Compatibility with Sleep Mode

Besides preventing message collisions and retransmissions, the most significant proposed solution to the low power requirement of the network is allowing nodes to periodically sleep [13]. This prevents nodes from indulging in idle listening which can consume unnecessary power. The Smart Stone Protocol's tight synchronization allows the nodes to sleep when global awareness is not required. One of the algorithms under development for the *Line in the Sand* application requires the nodes to be awake only during its own transmission slot, and the slot before, to receive the transmission of the neighboring node. This implements a cascading routing of information to the base station. While extending the SSP into a network-wide communication protocol we intend to incorporate a LEACH-like clustering mechanism [15]. The cluster head will be responsible for setting up TDMA schedules for the nodes that join its cluster, and will convey the *IDs* of the joining cluster nodes to the entire cluster. Subsequently, the cluster nodes will calculate their transmit times using equations such as 4.1 and 4.3. Each cluster node transmits its information during the calculated transmission slot and sleeps at all other times. This power saving mechanism allows the cluster head to be responsible for routing data to the base station or sinks, while the cluster nodes save their power. At the end of the round, new cluster heads are determined so that the high power tasks are evenly shared among the network nodes, while saving power by sleeping and minimizing collisions. Furthermore, in long term applications, events may not occur for days, during which the nodes do not transmit and remain in a low power mode. The sensing of disturbances may wake a certain node up, and cause it to transmit. This event resynchronizes the entire network and resumes synchronized operation at the first valid reception. In case of collisions on restart, the first transmission without interference will resynchronize the network.

PROTOCOL	*Precision*	*Piggy-backing*	*Complexity*	*Convergence Time*	*Network Size*
SSP (Thesis)	**50µs**	**YES**	**low**	**low**	**128+ SSN**
RBS [16]	1.86µs	NO	high	N/A	2-20
Romer [17]	3ms*	YES	low	N/A	Unknown
Mock et al.[18]	150µs	NO	high	low	Unknown
TDP [19]	100µs	NO	high	high	200 (*Sim*)
Ganeriwal et al.[20]	16.9µs	NO	low	Unknown	150-300(*Sim*)
ADP [22]	Unknown	NO	high	high	200-400(*Sim*)

Table 3.3 Performance Evaluation of Clock Synchronization Algorithms. *In a sparse ad-hoc network

3.4.3 Smart Stone Protocol (SSP) Summary

As highlighted in table 3.3, the SSP achieves a receiver-to-receiver synchronization precision of 50µs, and also synchronizes the sender to the receivers with a worst case skew of 100µs. This serves as an improvement on the protocols [16, 18] which use the broadcast property but leave the sender desynchronized from the network. The protocol is extremely low power, achieving synchronization within a single message, piggybacked on data transmissions without transmitting a single extra byte. The convergence time is the time taken to receive the first valid transmission. The protocol has been tested on the Smart Stone Network, adding nodes at random, removing nodes, and adding them back again without a single interruption in network functioning. Experiments were conducted to prove that the protocol in its current implementation can support clusters with up to 128 nodes. Finally, the Smart Stone network was equipped with MEMs microphones to serve as a sensor network. The network successfully shared information regarding acoustic vibrations sensed by the network nodes. Proposed extensions to the algorithm and advanced considerations are discussed in Chapter 4.

CHAPTER 4

4. Future Work

Contributions to the Smart Dust goal of developing low-power, low cost dust-sized wireless sensor nodes, to perform distributed processing to realize a particular sensor application, are being made by several research groups. One of the most fascinating aspects of the project is the possibility of contribution from almost every concentration in Electrical Engineering, and other engineering and scientific disciplines. Areas of active research dedicated to Smart Dust include

- Battery and power source development (example: Solar cells)
- Sensor Technology (example: MEMs)
- Fabrication Technology (example: 3D Integration)
- Circuit Design (example: Low power transceivers and ASICs)
- Mote Design (example: COTS wireless sensor motes)
- Synchronization Protocols
- MAC Protocols
- Routing Protocols
- Distribution Strategy

Each of the above mentioned points, however, is highly dependent on the specific application the WSN is designed to serve. A list of several prototyped applications is available in [43].

The Smart Stone Protocol (SSP) presented in Chapter 3 promises to be an excellent synchronization and medium access protocol for peer-to-peer networks, or for intra-cluster communication. Its robust, low power features have been designed with extension to the entire network in mind. Future work to develop the SSP into a network-wide synchronization and MAC protocol is proposed in this chapter.

4.1 Fine-tuning the Smart Stone Synchronization Protocol

The SSP synchronization precision between receivers has been proven to be within 50μs. RBS [16] experiments observed a maximum deviation of 53.4μs supporting our observation. However, Elson et al's observations were based on Berkeley Motes [8] running *TinyOS* as the operating system. TinyOS is a much more involved operating system than the assembly level program implementing the SSP on the Smart Stones. Faster timers running on the Smart Stones may be able to reveal more impressive experimental lower and upper bounds on the synchronization precision. As discussed in Chapter 1, algorithms synchronizing sender and receiver utilize the round trip time to estimate the message passing delay. In Chapter 3, we introduce an identifiable time T_{buff} on the sender that is closer to the time of arrival at the receiver. This time is the result of our software routines being closely related to hardware events, and the timing control obtained by programming the MCU in assembly language, as opposed to higher level languages such as *C*. T_{buff} removes the non-deterministic *send* time, and reduces the error in estimating *access* time, since this time occurs after the report is prepared, and sent over the UART. Further characterization of this time might supply the research community with a more accurate time to use for round trip estimation in methods like Romer's [17]. The second message sent in [17] can contain T_{buff}, instead of T_{prep}.

4.2 Increasing Smart Stone Cluster Size

The SSP in its current implementation based on *TIMER0*, an 8 bit timer, has been proven to support at least 128 nodes in a peer-to-peer network, or cluster. Though 128 nodes seems to be sufficient for a single cluster, the need to accommodate more nodes might arise. To accommodate 256 nodes, the slot size must be reduced from 26ms to 13ms. Experiments conducted demonstrated that the *localTime* count was the same before and after transmitting the report. This indicates that the 5ms switching time within the slot causes the entire slot processing time to be over 13ms. Programming the nodes to switch to transmit mode at the end of the previous slot, instead of the start of the transmitting slot may solve the problem. Keeping time over longer periods merits larger counters (64 bits suggested by Mock et al [35]). This would provide the SSP with larger frame sizes to accommodate larger number of nodes.

4.3 Incorporating external features into the SSP

Since the SSP achieves high precision synchronization at the smallest transmission overhead, it lends itself well to the incorporation of desired features present in other protocols. Though a precision of 50μs is sufficient for our TDMA slots, and acoustic vibration sensing tasks, applications which require a higher level of precision can utilize the RBS [18] technique of receivers sharing receive times to improve the synchronization precision. The sharing of receive times with respect to a particular node, during TDMA slots can be used to realize this. Our clock correction method involves some time discontinuity. Given the tightness of the synchronization, these discontinuities are small. Mock et al [18] suggest continuous clock synchronization to avoid this. This introduces complexity into the processing. Though not discussed in [35], backwards corrections achieved by slowing down the clock allow the MCU time to meet time triggered processing deadlines. A forward correction, however, involves speeding up the clock, and might not allow enough time for the MCU to complete the task within the time allotted. Though a scheduling mechanism can be developed to reschedule the events that lie in the correction interval, the continuous clock synchronization scheme can be used within the SSP, if desired.

4.4 Extending the SSP to a Network-wide Protocol

The most significant part of our future work will be directed towards extending the SSP into a network-wide synchronization and medium access protocol. This will be achieved by developing a routing protocol that lends itself to the SSP framework. The cluster nature of the current SSP implementation suggests a LEACH-like [15] routing protocol to be ideal to achieve the goal. As discussed in 1.3.3, the LEACH uses TDMA for intra-cluster communication, where the elected cluster head (CH) after an "advertisement phase" sends back the schedule for the nodes in its cluster. For the SSP, the schedule would be the values of *localTime* for each node to transmit at. To reduce the message size, the CH can simply send the IDs of the cluster nodes back (*node k, node p, ... node x*) and the nodes can compute their transmission slots by using the order of the nodes, and the total number of nodes. The transmission slots could be determined using the fixed slot size 26ms, using which the transmit times would be:

$$Txtime_i = slot\_length * (i-1) \tag{4.1}$$

$$frame\_length = slot\_length * N \tag{4.2}$$

where *i* is the index at which the node appears in the CH's report, and *N* is the total number of nodes. While this scheme is power efficient and reduces latency, a scheme of maintaining equal frame lengths throughout network clusters would be easier to distinguish rounds of the LEACH algorithm. For this case

$$Txtime_i = \frac{frame\_length}{N} * (i-1) \tag{4.3}$$

where *frame_length* is a constant, 3.3s in the current SSP implementation. If the base station is capable of communicating the real-time to any node in the network, schemes can be devised to synchronize CHs to each other when they broadcast information to the base station. Events can be reported after adjusting the cluster time values to the CH's local time.

4.5 Simulations and Implementation

Though the work in this thesis has gained strength by focusing on implementation of synchronization and MAC protocols on physical wireless sensor motes, extending the SSP to the entire network will require several simulations. Network Simulations can be used to gauge algorithm feasibility, simulate thousand of nodes, and determine the best values for parameters used in routing and MAC protocols. The PACT protocol [14] was simulated on GlomoSim [44], and LEACH [15] on network simulator ns[45]. *TinyOS* can be simulated using TOSSIM [46] which compiles directly from TinyOS code. Also, though the SSP has demonstrated excellent results on the Smart Stones, which have intentionally been built with limited hardware resources, testing the SSP on Berkeley Motes would greatly strengthen its appeal to the research community. The Berkeley Motes currently provide an excellent platform for researchers to compare their protocols.

4.6 Advanced Considerations

While tailoring existing concepts in communication protocols to serve WSNs proves an invaluable method to develop networking algorithms, “out of the box” approaches might hold the key to realizing the Smart Dust goal. The challenging problem of achieving low power message routing in WSNs might have solutions in everyday life. Mobile nodes traversing the network can serve as “postman nodes”, collecting messages from sources with intended destinations, and passing them to these destinations, if and when they are within transmission range. The use of mobile agents has also been stated by Tong et al. in [47]. A special case of the “postman node” is obtained by extended the concept to the third dimension. Technological advances in the field of Micro Air Vehicles (MAVs), small Unmanned Aerial Vehicles (UAVs) powered with transceivers, might serve the purpose of providing a two-hop routing path between nodes in very large scaled networks. The prospect of using UAVs for routing has also been explored by Gu et al. in [48].

4.7 Conclusions

I would like to thank everyone who makes it this far in this thesis for taking time to peruse this work. I hope this thesis piques the interest of newcomers, and motivates them to join the field of Wireless Sensor Network research. For those already part of the team, I sincerely hope this thesis has helped answer some of your questions, and in particular hope to see the Smart Stone Protocol developed into a robust network-wide synchronization and communication protocol, worthy of application on Smart Dusts!

REFERENCES

[1] J. Hill, M. Horton, R. Kling, L. Krishnamurthy, "The Platforms Enabling Wireless, Sensor Networks", *Communications of the ACM,* 47(6):41–46, June 2004.

[2] Distributed Surveillance Sensor Network. ONR SPAWAR Systems Center, San Diego. Available at (http://www.spawar.navy.mil/robots/undersea/dssn/dssn.html)

[3] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin, "Habitat Monitoring with Sensor Networks", *Communications of the ACM,* 47(6):34–40, June 2004.

[4] P. Johnson, D. Andrews, "Remote Continuous Physiological Monitoring in the Home", *Journal of Telemedicine Telecare,* 2(2):107–113, 1996.

[5] J. Werb, C. Lanzl, "Designing a Positioning System for Finding Things and People Indoors", *IEEE Spectrum,* 35(9):71–78, Sept. 1998.

[6] J. Kahn, R. Katz, K. Pister, "Next Century Challenges: Mobile Networking for SmartDust", *Proc. Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking,* pp. 271–278, 1999.

[7] G. Pottie, W. Kaiser, "Wireless Integrated Network Sensors", *Communications of the ACM,* 43(5):51–58, May 2000.

[8] D. Estrin, R. Govindan, "Next Century Challenges: Scalable Coordination in Sensor Networks," *MobiCom'99*, Seattle, WA, pp.263-270, Aug. 1999.

[9] R. Shah and J. Rabaey, “Energy Aware Routing for Low Energy Ad hoc Sensor Networks”, *IEEE Wireless Communication Conference (WCNC),* pp. 350–355, March 2002.

[10] Mark Stemm and Randy H Katz, “Measuring and reducing energy consumption of network interfaces in hand-held devices,” *IEICE Transactions on Communications*, vol. E80-B, no. 8, pp. 1125–1131, Aug. 1997.

[11] I. Akyildiz, W. Su, Y. Sankarasubramanian, E. Cayirci, “Wireless Sensor Networks: A Survey”, *Computer Networks,* 38(4):393–422, March 2002.

[12] K. Sohrabi, J. Gao, V. Ailawadhi, G. Pottie, “Protocols for self-organization of a wireless sensor network”, *IEEE Personal Communications*, pp. 16-27, October 2000.

[13] W. Ye, J. Heidemann, D. Estrin, “An Energy-Efficient MAC Protocol forWireless Sensor Networks”, *Proc. of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002),* pp. 1567–1576, June, 2002.

[14] G. Pei, C. Chien, “Low power TDMA in large wireless sensor networks,” *Military Communications Conference (MILCOM 2001)*, volume 1, pp. 347–351, Vienna, VA, October 2001.

[15] W. Heinzelman, A. Chandrakasan, H. Balakrishnan, "Energy-efficient communication protocol for wireless sensor networks," *Proceeding of the Hawaii International Conference System Sciences,* Hawaii, January 2000.

[16] J. Elson, L. Girod, D. Estrin, “Fine-Grained Network Time Synchronization using Reference Broadcasts”, *Proc. Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002),* Vol 36, pp. 147–163, 2002.

[17] K. Romer, “Time Synchronization in Ad Hoc Networks”, *Proc. ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc ’01),* pp. 173–182, Oct. 2001.

[18] M. Mock, R. Frings, E. Nett, S. Trikaliotis, “Continuous Clock Synchronization in Wireless Real-time Applications”, *Proc. 19th IEEE Symposium on Reliable Distributed Systems (SRDS-00),* pp. 125–133, Oct. 2000.

[19] W. Su, I. Akyildiz, “Time-Diffusion Synchronization Protocols for Sensor Networks”, *IEEE/ACM Transactions on Networking*, 2005, in press.

[20] S. Ganeriwal, R. Kumar, S. Adlakha, M. Srivastava, “Network-wide Time Synchronization in Sensor Networks”, *Technical Report, Networked and Embedded Systems Lab,* Elec. Eng. Dept., UCLA, 2003.

[21] S. PalChaudhuri, A. Saha, D. B. Johnson, “Probabilistic Clock Synchronization Service in Sensor Networks”, *Technical Report TR 03-418*, Department of Computer Science, Rice University, 2003.

[22] Q. Li, D. Rus, “Global Clock Synchronization in Sensor Networks”, *Proc. IEEE Conf. Computer Communications (INFOCOM 2004),* Vol. 1, pp. 564–574, Hong Kong, China, Mar. 2004.

[23] M. Kohvakka, M. Hännikäinen, T. D. Hämäläinen, “Wireless sensor network implementation for industrial linear position metering,” *Proc. IEEE Euromicro Conf. on Digital Systems Design,* Aug. 2005, Portugal, accepted.

[24] J. Elson, K. Romer, “Wireless Sensor Networks: A New Regime for Time Synchronization”, *Proc. First Workshop on Hot Topics In Networks (HotNets-I),* Princeton, New Jersey. Oct. 2002.

[25] B. Sundararaman, U. Buy, A. Kshemkalyani, “Clock synchronization in wireless sensor networks: A survey,” *Ad-Hoc Networks* 3(3) (2005) p.281--323

[26] D. Culler, W. Hong, “Special issue on Wireless Sensor Networks” *Communications of the ACM,* 47(6):30–34, June 2004.

[27] IEEE, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std. 802.11,* November 1997.

[28] K. Pister, J. Kahn and B. Boser, "Smart Dust: Wireless Networks of Millimeter-Scale Sensor Nodes", *Highlight Article in 1999 Electronics Research Laboratory Research Summary.*

[29] J. Kahn, R. Katz and K. Pister, "Mobile Networking for Smart Dust", *ACM/IEEE Intl. Conf. on Mobile Computing and Networking (MobiCom 99)*, Seattle, WA, August 17-19, 1999.

[30] http://ww1.microchip.com/downloads/en/DeviceDoc/30487c.pdf

[31] http://www.linxtechnologies.com/documents/TR-xxx-SC_Data_Guide.pdf

[32] https://www.barebonespcb.com/!BB1.asp

[33] http://www.cadsoft.de/

[34] http://www.inf.ethz.ch/˜kasten/research/bathtub/energy_consumption.html

[35] M. Mock, R. Frings, E. Nett, and S. Trikaliotis, "Clock Synchronization for Wireless Local Area Networks", *12th Euromicro Conference on Real-Time Systems (ECRTS)*, Stockholm, June 2000.

[36] H. Kopetz and W. Schwabl, "Global time in distributed real-time systems." *Technical Report 15/89*, Technische Universit¨at Wien, 1989.

[37] J. Elson and D. Estrin, "Time synchronization for wireless sensor networks", *Proceedings of the 15th International Parallel and Distributed Processing Symposium (IPDPS-01)*. IEEE Computer Society, April 23–27 2001.

[38] W. Merrill, L. Girod, J. Elson, K. Sohrabi, F. Newberg, and W. Kaiser, "Autonomous Position Location in Distributed, Embedded, Wireless Systems",

Proceedings of IEEE CAS Workshop on Wireless Communications and Networking, Pasadena, CA, September 2002.

[39] H. Wang, L. Yip, D. Maniezzo, J.C. Chen, R.E. Hudson, J.Elson, and K.Yao, "A Wireless Time-Synchronized COTS Sensor Platform Part II–Applications to Beamforming", *Proceedings of IEEE CAS Workshop on Wireless Communications and Networking*, Pasadena, CA, September 2002. http://lecs.cs.ucla.edu/Publications.

[40] www.trxsystems.com

[41] M. Gergeleit, and H. Streich, "Implementing a Distributed High-Resolution Real-Time Clock using the CAN-Bus", *1st international CAN-Conference 94*, Mainz, 13.-14.09., CAN in Automation e.V., Erlangen 1994.

[42] P. Veríssimo, and L. Rodrigues, "A posteriori Agreement for Fault-tolerant Clock Synchronization on Broadcast Networks", *22th Int. Symp. on Fault-Tolerant Computing* Boston, July 1992.

[43] M. Kuorilehto, M. Hannikainen, T. Hamalainen, "A Survey of Application Distribution in Wireless Sensor Networks", *EURASIP Journal on Wireless Communications and Networking*, Vol. 5, pp: 774–788, 2005.

[44] M. Takai, L. Bajaj, R, Ahuja, R. Bagrodia and M. Gerla, "GloMoSim: A Scalable Network Simulation Environment," *Technical report 990027*, UCLA, Computer Science Department, 1999.

[45] http://www-mash.cs.berkeley.edu/ns/ UCB/LBNL/VINT Network Simulator - ns (Version 2), 1998.

[46] http://www.tinyos.net/

[47] L. Tong, Q. Zhao, and S. Adireddy, "Sensor networks with mobile agents," *Proc. IEEE MILCOM'03*, Boston, MA, October 2003.

[48] D. L. Gu, G. Pei, H. Ly, M. Gerla, and X. Hong, "Hierarchical routing for multi-layer ad-hoc wireless networks with UAVs," *MILCOM'00*, Los Angeles, CA, October 2000.