Time synchronization in wireless sensor networks

Bertold van Voorst
Faculty of Electrical Engineering, Mathematics and Computer Science
University of Twente, the Netherlands
b.g.vanvoorst@student.utwente.nl

ABSTRACT

Wireless Sensor Networks (WSNs) consist of numerous small sensors that are wirelessly connected to each other. In these networks, time synchronization is an important issue. Correct time-stamping of events is crucial for data processing.

Time synchronization protocols that are used for synchronization in traditional wired networks are not applicable in WSNS. This paper provides a survey of a number of synchronization protocols that have been specified for use in WSNs. Moreover, a comparison of these protocols is provided in order to identify the most efficient one.

Keywords

wireless sensor network, time synchronization

1. INTRODUCTION

Recently, small, low power sensors with embedded processors and radios have become available [EST02]. Moreover, a wide range of application areas, including military, environment, health, home and other commercial areas [AKY02] has been identified that could very much profit from using a network of such sensors. As a result of this, wireless sensor networks (WSNs) have become an important and growing research area. A WSN consists of numerous cooperating sensors, connected to each other in a wireless network.

In WSNs, time synchronization is an important issue. Precise and synchronized time is needed for several reasons. For example, an exact and synchronized clock is necessary to determine the right chronological order of events. A lack of synchronization may lead to incorrect time-stamping of readings. As a base station collects the sensor data, it may not be reordered right [DAI04]. Further, accurate time is needed for applications as time-of-flight measurements, low power TDMA radio schedules and duplicate detection recognition [ELS03].

These many applications make time synchronization an important issue in WSN research. In the last few years, numerous time synchronization protocols and methods have been proposed based on different approaches. In this paper, a comparison between these protocols and methods is provided, in order to find the most efficient one.

The organisation of the remainder of this paper is as follows. In section 2 an overview of the different aspects and characteristics of WSNs is given. Section 3 gives an overview of existing time synchronization protocols in traditional networks and discusses why these protocols are not suitable for WSNs. An overview of proposed time synchronization methods for WSNs is given in section 4. These methods are compared to each other in section 5. Finally, in section 6, conclusions and suggestions for future work are given.

2. WIRELESS SENSOR NETWORKS

This chapter gives an overview of the different aspects and characteristics of WSNs.

2.1 Sensors

A WSN consists of a number of sensors, ranging from a few to thousands of sensors that are wirelessly connected to each other or to a base station. For wireless communication, a number of communication technologies can be used. Examples of such technologies are radio, infrared light, laser and inductive and capacitive coupling. Radio communication is currently the most common, since no line of sight is required and communication over medium ranges is possible with acceptable power consumption [ROM04]. In networks where line-of-sight is possible, infrared or laser may be used to achieve an even lower power consumption.

Many different sensors exist, which makes it possible to do a wide variety of different measurements. A few examples from a long list are the ability to sense light, temperature, humidity, acceleration, chemical vapors, gas concentrations and noise levels [MAI02].

Depending on the application, the size of a single sensor may vary from the size of a shoebox down to a few millimeters. Also, the cost of a single device may range from a few hundred Euros for networks of few powerful nodes, to just a few cents for networks of many simple nodes [ROM04]. These devices are very limited in the amount of energy they can store or harvest from their environment. Therefore, energy efficiency is a very important issue in WSN development. Manually replacing batteries or doing maintenance on sensors is undesired and often also impossible to perform. Most sensors thus require untethered (wireless), autonomous operation.

2.2 Dynamic operation

In contrast to traditional wired networks, most WSNs are highly dynamic. The system must continuously adapt to changes in the environment. The sensors may be deployed randomly, for example by dropping them from an aircraft into an area of interest. Over time, sensors may fail and batteries get depleted. The WSN must adept to these changes to prevent network failure [HIL03].

Mobility and scalability are important design principles. A sensor or an entire WSN may physically be moved to another area or be combined with another WSN. In this type of situations WSN self-configuration (i.e., auto-configuration) is often required because static configuration is difficult to be performed.

2.3 Network topology

The network topology of WSNs may vary. Traditional wired networks are connected to each other through switches and routers. As a result, some sort of infrastructure hierarchy is always present. In WSNs, such an infrastructure hierarchy is absent.

The most important factor is the diameter of a network, which is defined as the maximum number of hops between any two end-nodes in the network [ROM04]. The diameter depends on the size of the network, but also on the communication range of the nodes.

The simplest form is a single-hop network, where every node can directly communicate with every other node in the network. This is only possible in small networks or when nodes have a wide communication range. A more common topology is a multi-hop ad hoc network topology [MEG01], in which data packets are forwarded via multiple nodes in order to reach the destination.

2.4 Operating system

The operating system that runs on most sensors is the TinyOS [GAY05]. This operation system is specially designed for WSNs. It was built with an emphasis on reacting to external events and extremely low power operation. Examples of other operating systems that can be run on wireless sensor nodes are an embedded version of Linux operating system [MAI02] and a multithreaded μ-OS operating system [AKY02].

3. TIME SYNCHRONIZATION IN TRADITIONAL NETWORKS

Network time synchronization is not a new research topic, though it is in the area of WSNs. Much research has been done on how to transfer time information in order to synchronize clocks in a network.

In this chapter, an overview of time synchronization protocols in traditional networks is given. Furthermore, the differences in time synchronization between traditional networks and WSNs are explained.

3.1 Existing time synchronization mechanisms in traditional wireless/wired networks

The most common time synchronization protocol that is used in traditional networks is the Network Time Protocol (NTP) [MIL92]. A client using NTP connects to a server to synchronize time. The time given by the server is corrected with half the measured round trip time. This results in a synchronization with an error of at most a few milliseconds. NTP is a two-way time synchronization method, in which a timestamp is sent in two directions. To improve the accuracy, there is also a method available that sends double packets [GOT02].

The Simple Network Time Protocol (SNTP) [MIL96] is a simplified version of the NTP. This protocol allows operation in a stateless remote-procedure call (RPC) mode. It is less complex then the full NTP, which makes it easier to implement, but decreases accuracy. SNTP is therefore useful in systems where the performance of a full NTP implementation is not significant.

Where NTP and SNTP are based on a hierarchical structure, the Classless Time Protocol (CTP) [GUR03] uses a peer-to-peer approach. Using this protocol, a node sends and receives packets only to and from its direct neighbors. The CTP uses the same format and number of network messages as the NTP. According to [GUR03], the performance of the CTP is better than the performance of the NTP without an increase of complexity.

The Global Positioning System (GPS) can also be used for time synchronization [BER00, JUA02]. The accuracy can be high, with an error of less than a millisecond. This is however hardly used in computer networks. GPS requires a clear view on the sky for continuous reception of multiple satellites, which is hard to achieve indoors. Also, specific hardware is required to receive the GPS signals which makes GPS an expensive solution for networks with many nodes.

3.2 Differences between time synchronization mechanisms in traditional networks and WSNs

It would be easy if these existing time synchronization protocols could be directly used in WSNs. This is however not possible. Many requirements on which these protocols are based do not hold for WSNs.

3.2.1 Energy awareness

One of the most important issues in WSNs is energy efficiency. For time synchronization protocols in traditional wired networks, using the CPU power listening to the signaling messages from the network and supporting occasional transmissions have a negligible impact on the operation of a communicating node. Traditional time synchronization protocols are based on the following assumptions. NTP assumes that the CPU is always available and it listens to the network all the time. These assumptions do not hold for WSNs [POT00]. WSN nodes often have slow CPU's that are operating most of the time in an idle power mode. Even though processing power costs decrease with time, there are limits that keep it relatively expensive for nodes in a WSN. Listening to, receiving from and sending to the network also cost a significant amount of energy, compared to the overall system budget. This could very difficult be supported by wireless sensor nodes that are typically using batteries, which have a limited energy lifetime and power and are difficult to be replaced.

3.2.2 Ad hoc network topology

Most time synchronization protocols are based on a hierarchical network infrastructure. The servers on the highest level, often referred to as stratum 1 servers, synchronize with each other using technologies as GPS. Nodes in these hierarchical networks will usually have a stratum 1 server only a few hops away. In WSNs on the other hand, a hierarchical infrastructure is usually absent. Even if we can create such a hierarchy, most nodes will be far away from it. Nodes that are far away from the stratum 1 server will be poorly synchronized. In WSNs, this is especially a bad situation, since nodes that are close to each other often require the most precise synchronization [ELS02]. This is illustrated in figure 1. Nodes A, B and C are far away from the stratum 1 server, but close to each other. In a scheme as NTP, C will choose either A or B as its synchronization source. The synchronization error with the opposite neighbor will be large. This is caused by the fact that the path to the stratum 1 server is dependent on the synchronization source that is chosen. If C synchronizes with A, the error with B will be large; if it synchronizes with B, the error with A will be large.

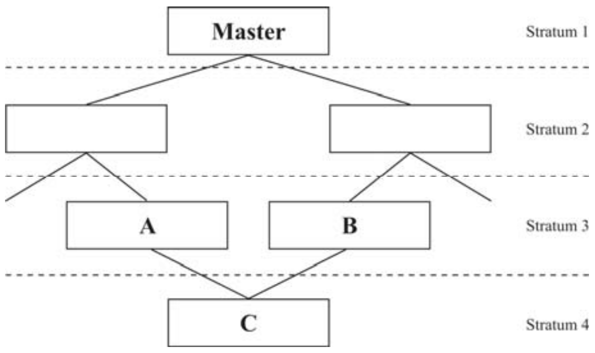

Figure 1. Nodes that are far away from the stratum 1 server will be poorly synchronized, which is especially a bad situation in WSNs.

Solving this problem by creating many stratum 1 servers within a WSN is not an option. Equipping nodes with GPS receivers is difficult, since it is expensive in energy consumption as well as monetary and it requires a line of sight to the GPS satellites.

3.2.3 Dynamic infrastructure

Traditional networks are based on a static infrastructure. For time synchronization, nodes are manually configured to connect to certain time synchronization servers. Although it is possible to use statistical information to decide which servers to use, nodes are still dependent on their initial configuration. Because of the highly dynamic nature of WSNs, such a static configuration is not possible. Moreover, the need for unattended operation makes a manual configuration of individual nodes highly undesirable.

3.2.4 One-way synchronization

Most existing time synchronization protocols use two-way methods, meaning that information is sent in two directions. Ping [PIN03] argues that two-way methods are not suitable in WSNs. If the path between the node and the time synchronization server is reciprocal or symmetric, the one-way delay can be estimated as half the round-trip time. In WSNs, this path is often not reciprocal, which makes it difficult to estimate the delay. One-way methods for time synchronization are therefore better candidates for WSNs.

3.2.5 Tunable accuracy

Time synchronization protocols in traditional networks are designed to achieve the highest accuracy possible. The higher the required accuracy, the higher the resource requirements. In WSNs, it may be useful to make a trade-off between accuracy and resource requirements [TUL04]. This way, a time synchronization protocol may use less energy or use a less advanced CPU when high accuracy is not required.

4. TIME SYNCHRONIZATION METHODS FOR WSNS

In the last few years, numerous time synchronization protocols for WSNs that are using different approaches have been proposed. A number of these protocols are summarized in this chapter.

4.1 Reference Broadcast Synchronization

Reference Broadcast Synchronization (RBS) [EGE02, ELS03] is a time synchronization protocol in which nodes send reference beacons to their neighbors, by making use of the broadcast possibility of the network. These beacons do not contain a timestamp. Receivers use the arrival time of these beacons as points of reference for comparing their clocks. A beacon can be sent by any node in the network, thus no special nodes are needed.

The simplest form of RBS is executed in three steps:

1. A node broadcasts a reference beacon.
2. Each node that receives the beacon, records its arrival time according to the node's local clock.
3. The nodes exchange their observations. Using this information, each node can compute its offset to any other node.

Using this protocol, a relative timescale can be formed. It is also possible to synchronize with an external timescale. For example, consider a GPS receiver connected to one of the nodes in the network. This GPS node is treated exactly as other nodes in the network. The offset from any node to the GPS node can then be used to calculate the absolute time.

The description of RBS given so far assumes that the network is fully connected, which means that a broadcasted beacon is received by all nodes. In reality, this will hardly ever be the case. Figure 2 explains how this problem is handled. Nodes a and b send out beacons that create neighborhoods, i.e., groups of nodes that are in the transmission range of a single beacon node. Node a has four other nodes in its neighborhood. Node 4 receives signals from both a and b, and can act as a "gateway" to relate nodes in each neighborhood to each other.

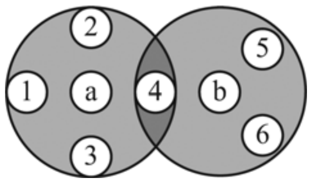

Figure 2. A simple multihop network. Nodes a and b send beacons; node 4 acts as a "gateway".

To illustrate how this works, imagine we would like to relate the times of two events that happen in the network of figure 2. Node 1 observes event E_1 2 seconds after hearing the beacon signal B_a ($E_1 = B_a + 2$). Node 6 observes event E_6 9 seconds after hearing beacon signal B_b ($E_6 = B_b + 9$). Node 4 has heard beacon signal B_a 5 second prior to B_b ($B_a + 5 = B_b$). Given these constraints, we can easily calculate the time that has passed between E_1 and E_6:

$$E_1 = B_a + 2$$

$$E_6 = B_b + 9$$

$$B_a + 5 = B_b$$

$$\rightarrow \quad E_1 + 12 = E_6$$

Event E_6 has thus occurred 12 seconds after the occurrence of event E_1.

The precision can be improved by sending more than one reference beacon. The offset between two nodes is then computed as the average of the offsets of each single node. Clock skew, caused by the fact that clocks never run at exactly the same rate, can also be corrected. This clock skew can be estimated from the time that has passed between hearing multiple reference beacons.

4.2 Probabilistic extension to RBS

Guarantee can not be provided when delays can be unbounded or messages can get lost. This may be a problem when RBS is implemented. PalChaudhuri et al. [PAL04] therefore present a probabilistic extension to RBS. This extension is based on the possibility of RBS to send multiple beacon messages to improve accuracy.

For this protocol we need to specify the maximum synchronization error and the confidence probability. The minimum number of messages and the synchronization interval are then derived from this specification. This is possible, because the error among receivers has a normal distribution.

The following steps are taken to synchronize a broadcast neighborhood:

1. A sender broadcasts *n* reference beacons to its neighbors at fixed intervals.
2. Each node that receives the beacons, records the arrival times of each beacon according to the node's local clock. This data is plotted and a line is derived from it. The slope of this line approximates the relative clock skew between sender and receiver.
3. All receivers send their graph back to the sender.
4. The sender combines all graphs and broadcasts a message containing its relative clock skew to all the receivers.
5. Every receiver can now calculate its own clock skew and clock offset relative to all other nodes in the neighborhood.

For multihop networks, the protocol is also extended. RBS as described in section 4.1 assumes that there is always a "gateway" node that receives beacons from both neighborhoods to link them together. In this extension, it is not necessary to make such an assumption. Synchronization outside the neighborhood of a sender is handled by making nodes senders themselves once they get synchronized. This may cause a chain reaction and flood the network with synchronization messages. To avoid this, a sender will only broadcast a beacon when a synchronization request is made by a receiver in its neighborhood.

4.3 Timing-sync Protocol for Sensor Networks

The Timing-syncs Protocol for Sensor Networks (TPSN) [GAN03] synchronizes time in a sensor network by first creating a hierarchical structure and then synchronizing nodes along this structure. In this way, a global timescale is established throughout the network.

A hierarchical topology is created by assigning a level to every node in the network. Only one node is assigned level 0, which will be the master node. Every other node at level *i* can communicate with at least one other node at level *i - 1*. The creation of this hierarchical topology is called the level discovery phase, and occurs when the network is deployed. This phase is initiated by the level 0 node, by broadcasting a *level_discovery* packet, containing the level and identity of the sender. Every node that receives this packet, assigns itself a level one greater than the level they received. After this, they broadcast a new *level_discovery* packet which contains their own level and identity. Eventually, all connected nodes will have a level assigned. To prevent network flooding, a node neglects all *level_discovery* packets once it has a level assigned.

It may be possible that a node does not receive a *level_discovery* packet. When a node has not been assigned a level after a certain period, it timeouts and broadcasts a *level_request* message. Nodes that receive this *level_request* message, reply to this by sending their own level.

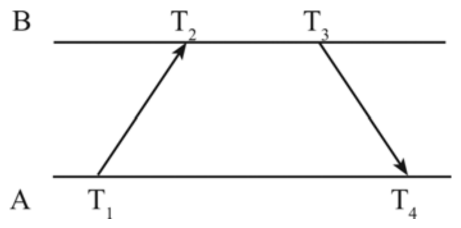

Figure 3. Pair wise synchronization between A and B.

When a hierarchical structure is established, the synchronization phase can be started. In this phase, pair-wise synchronization is performed along the edges of the structure. This pair-wise synchronization works as shown in figure 3. Here, node A synchronizes with node B. T_1 and T_4 represent time as measured by node A, T_2 and T_4 represent time as measured by node B.

1. At T_1, node A sends a *synchronization_pulse* to node B.
2. Node B receives this packet at T_2 and sends an *acknowledgement* packet back to node A at T_3, containing T_1, T_2, and T_3.
3. Node A can now calculate the clock skew using the following formula:

$$\Delta = \frac{(T_2 - T_1) - (T_4 - T_3)}{2}$$

The synchronization phase is initiated when the level 0 node broadcasts a *time-sync* packet. When the level 1 nodes receive this packet, they will wait for some random time before starting pair-wise synchronization with the level 0 node. This randomization is to avoid congestions. The nodes on level 2 will hear this message exchange. After a random delay they synchronize with the level 1 node. This way the synchronization works down the hierarchy until all nodes have been synchronized.

Due to the dynamic nature of wireless sensor networks, it may happen that a situation occurs where a node at level *i* can not communicate to a node at level *i - 1*. In this case, a node will not receive an *acknowledgement* back when it sends a *synchronization_pulse*. This node can then broadcast a *level_request* message. Assuming that the node is still connected to the network, it will be assigned a new level. If this happens at level 1, it means that the single level 0 node can no longer be reached. The level 1 nodes will now run a leader election algorithm to determine the new level 0 node.

4.4 Time-Diffusion Synchronization Protocol

The Time-Diffusion Synchronization Protocol (TDP) [SU05] is a protocol that allows synchronization of a whole sensor network. Initially, an election protocol is run to select a set of master nodes. These masters then send a broadcast message, containing their local time. All receivers send back a message, allowing the masters to calculate the round-trip times as in Figure 3. The average of these round-trip times is used to estimate the one-way delay between the master and its neighboring nodes.

The masters now broadcast the one-way delay time and the standard deviation. The receiving nodes store this information. Then, they run the election protocol and a number of them become masters themselves. The average delay time and standard deviations are accumulated along the path. This procedure is repeated until a number of hops from the initial masters is reached. This number depends on the implementation of the protocol and can differ per network.

Finally, all nodes have received the total delay time and standard deviation from one or more masters. Using these values, every node calculates a new time value and adjusts its clock.

After all nodes have updated their clocks, new masters are selected and the procedure is started from the beginning. Running this procedure multiple times will allow the nodes to converge to an equilibrium time. The clock differences between neighboring nodes will be small.

4.5 TSync

TSync [DAI04] is a bidirectional time synchronization service that offers both a push mechanism for synchronizing the whole network as well as a pull mechanism for on-demand synchronization of a single node.

This protocol uses the ability of node radios to communicate over multiple channels for frequency diversity. By doing this, it is possible to reduce packet collisions and prevent jamming. All time-critical packets are sent through a dedicated channel, called the clock channel.

4.5.1 Push: Hierarchy Referencing Time Synchronization Protocol

The Hierarchy Referencing Time Synchronization Protocol (HRTS) can synchronize a whole network. We assume there is a master node or base station that starts the synchronization procedure. The synchronization between two nodes works as shown in Figure 3. The following steps are executed.

1. The master node (A) broadcasts a *sync_begin* message containing its local time T_1 and the name of one of the receivers, node B in this example.
2. All receivers record the time T_2' that this message was received. Only the node that was specified by the master (node B in this example), sends a message over the clock channel back to the master, containing timestamps T_2 and T_3.
3. The master node now knows the values of T_1, T_2, T_3 and T_4. It calculates Δ and broadcasts a message containing T_2 and Δ.
4. All nodes now compare the value of T_2 with their local value T_2' that was recorded in step 2. They can calculate the offset δ between their local clock and the clock of node B as:

   $$\delta = T_2 - T_2'$$

   The clock is now corrected as:

   $$T = local\_time + \Delta + \delta$$

5. The synchronized nodes now become masters themselves and broadcast a *sync_begin* messages to synchronize more distant nodes.

As the synchronization ripple is going through the network, a hierarchy is constructed. Each node is assigned a level based on its distance to the master node, as in the TPSN protocol. This level is used by nodes to avoid being synchronized with a less accurate source. Once a node is synchronized and has been assigned a level, it will ignore *sync_begin* messages from nodes with a higher level number. This way, it is possible to use multiple master nodes.

In step 1, we assume that the master node knows its neighbors, since the name of one of the neighbors is specified in the *sync_begin* message. It may be necessary to run a neighbor discovery procedure in advance.

4.5.2 Pull: Individual-based Time Request Protocol

A situation may occur where a single node needs synchronization, but a full network synchronization is unnecessary. In these cases the Individual-based Time Request Protocol (ITR) may be used.

1. When a node n_1 needs to be synchronized, it sends a query to its parent in the hierarchy. The request is forwarded until it reaches a master node.
2. The path to the master node is now known, and all nodes along the path switch to the clock channel.
3. The node n_1 now sends the actual synchronization request at the clock channel which is forwarded to the master, also at the clock channel.
4. The master now sends the time back to node n_1, again at the clock channel. Upon receiving the time, n_1 can adjust its clock.

4.6 Lightweight Tree-based Synchronization

The Lightweight Tree-based Synchronization protocol (LTS) [GRE03] also consists of a push mechanism and a pull mechanism.

4.6.1 Push mechanism

The push mechanism and the TPSN are very much alike. In the first phase, a tree hierarchy is constructed. In the second phase, nodes synchronize to their parents by means of pair wise synchronization (figure 3). An algorithm is given to calculate the minimum synchronization frequency that is needed to achieve a specified precision.

4.6.2 Pull mechanism

This mechanism uses a tree hierarchy in the network, though it does assume that there are one or more master nodes to synchronize with. When a node *n* determines that it needs to be synchronized, it sends a synchronization request to a master node. The routing protocol for this procedure is not specified. In order for *n* to synchronize with a master node, all nodes on the path must be synchronized using pair wise synchronization.

There are a few enhancements possible when using this mechanism. When a node needs synchronization, it can first check its neighbors for pending synchronization requests. Another possible enhancement is path diversification. When a node *n* requests synchronization, all nodes on the path to the master node are synchronized as well. Choosing a path along nodes that will otherwise soon need synchronization for themselves can prevent extra synchronizations in the near future.

5. COMPARISON BETWEEN TIME SYNCHRONIZATION METHODS FOR WSNS

The methods that are listed in chapter 4 will be compared here. In order to do this, some criteria are given first.

5.1 Accuracy

The accuracy of a synchronization protocol is one of the most important criteria. The higher the accuracy of a protocol, the more precise the clocks in a WSN will be synchronized. Delays in message delivery lead to synchronization errors. There are four sources defined that cause these errors [EGE02, GRE03, GAN03, MAR04].

- **Send time**. This is the time it takes an application to send a network message. This includes the execution of system calls and access time to the network interface.
- **Access time**. When a network interface of a node is ready to send a message, it must wait for access to the transmit channel before actually sending the message. This delay depends on the current network load.
- **Transmission time**. The time it takes for the sender to send the message. This time depends on the speed of the radio.
- **Propagation time**. This is the time it takes the radio signal to travel through the air from the sender to the receiver. This depends on the distance between the sender and the receiver and on the propagation speed of the wireless medium.
- **Reception time**. The time it takes for the receiver to receive the message. This is similar to the transmission time.
- **Receive time**. Analogue to send time, it also takes some time for a network interface to process a received message and to send it to the application.

In order to gain a high accuracy, we must remove delay uncertainties as much as possible. The protocols that are explained in section 4 use different methods to accomplish this.

In pair wise synchronization where symmetric links are used, the error caused by propagation time is negligible. The propagation time is the same in both directions between two nodes. This assumption is true as long as nodes do not move very fast.

In TPSN and LTS, the send time error is minimized by using time-stamping packets at the MAC-level instead of at the application level. That means that the timestamp is added to the packet when it is ready to be transmitted. The receive time in TPSN is also minimized by time-stamping received messages at the MAC-level. LTS however, does not do this. This solution of time-stamping at the MAC-level depends on the sensor hardware support.

TSync uses a method to reduce access time. All time-critical packets are sent through a dedicated channel. The network load on this channel is very low because it is only used to send time-synchronization messages. On such a sparsely used channel, packet collisions and jamming will occur less often.

RBS eliminates both send time and access time. The beacon message that is initially sent does not contain a timestamp, nor is it important when exactly the message was sent. In contrast to pair wise synchronization methods, propagation time does account for a considerable error. The beacon message is sent in one way, and thus the propagation time can be large, depending on the distance between the nodes.

The accuracy of all protocols that are mentioned in this paper have already been tested by their developers. These tests are described in the papers in which the protocols are proposed, see [EGE02], [PAL04], [GAN03], [SU05], [DAI04] and [GRE03]. The synchronization errors that were measured, vary between 20 μs and 30 μs. However, these test results can not be compared directly to each other. The testing environments and circumstances are different in all of the tests. For example different hardware was used and the WSNs used for testing varied in size. There are no protocols that perform much better or much worse than others.

5.2 Energy efficiency

As mentioned before in this paper, energy efficiency is very important in WSNs. There are two actions that require the most energy consumption: using the CPU and transmitting data.

The calculations that must be made when using the described protocols are fairly simple. Only a few additions and a single division are necessary for clock offset calculation. These calculations are mostly the same in all protocols.

A more significant difference between the protocols can be found in data transmission. We can make an estimate of the number of messages that must be transmitted to synchronize a network.

For simple pair wise synchronization two messages must be transmitted to synchronize one node with another. For TPSN and LTS one extra message per node is needed to create the hierarchy, and another one to send the *time-sync* packets. In TSync the number of messages is reduced. Because masters broadcast their synchronization messages, multiple nodes can be synchronized while only three messages are transmitted. The first message is the *sync_begin* message broadcasted by the master. Next a reply from only one node is required, and finally T_2 and Δ are broadcasted to all nodes in the transmission range. The actual number of nodes that can be synchronized at once depends on the density of the network and the transmission range of the node's radio. This protocol will therefore be very energy-efficient in dense networks, while in networks with low node density it will need more messages per node.

When RBS is used, synchronization starts by a node broadcasting a reference beacon. After receiving this beacon, receiver nodes exchange their observations. It is not specified how this is done exactly, but it seems reasonable that each receiver broadcasts the local time at which it received the beacon. Synchronization is thus possible with a minimum of only one beacon message and one message per receiver. However, when more accuracy is needed or the extension to RBS is used, more beacons must be sent. This may also lead to more messages for observation exchange between receivers.

For TDP, a master broadcasts a message to all nodes in its range, next all receivers send a reply back and finally the master broadcasts another message. In addition to this, the initial master election algorithm that must be run, will need extra message exchanges. This number of messages depends on the chosen algorithm. The synchronization procedure must be run for several times to allow the node's clocks to converge to an equilibrium time. When a network is deployed, it will take about 7 synchronization rounds to reach an acceptable level of synchronization.

Another energy problem may occur when the load of a few nodes is much higher compared to other nodes. These nodes will consume their energy faster, causing their batteries to get exhausted earlier than batteries of other nodes. The chance that this problem will occur is especially big when using pull mechanisms like the ones in TSync and LTS. When a single node requires synchronization, it must someway require action from a master. When *n* nodes use the pull mechanism, the master node is involved in a synchronization procedure *n* times, while each client node is involved just one time.

In RBS this problem may occur at the gateway nodes. TDP solves this problem by regularly running an election protocol that elects different masters. The problem does not occur in TPSN, because all nodes send the same messages in the synchronization phase.

5.3 Tuning

As mentioned in section 3, tunable accuracy is a useful property of a synchronization protocol for WSNs. When a protocol supports tuning, it is possible to make a trade-off between accuracy and energy consumption.

The most simple form of tuning is adjusting the interval at which the clocks in a network are synchronized. This is possible with all protocols. This only helps to reduce the time that the clocks of different nodes will diverge from each other. The minimum synchronization error cannot be reduced in this way. In addition, some protocols have more advanced tuning options.

The possibility in RBS to send multiple beacon messages is used in the extension for RBS [PAL04]. This extension offers an algorithm that allows for specification of the maximum allowed synchronization error. The number of beacon messages will be based on this maximum allowed error. A lower allowed error will lead to more messages and thus higher energy consumption.

When using TDP, the synchronization procedure must be run several times for the nodes to reach an equilibrium time. The interval between these rounds determines how fast equilibrium is reached. In time, this also determines the accuracy that can be reached.

5.4 Post-facto synchronization

Post-facto synchronization [ELS01] is a method for further reducing energy consumption. In this scheme, nodes are normally unsynchronized. When an event occurs, the node records the time according to its local clock. The node must now synchronize its time to know its clock offset. When this has happened, it can re-calculate the time at which the event occurred with respect to the synchronized time. This method allows nodes to turn off their power, waiting only for certain events to happen.

Post-facto synchronization is not applicable with all protocols. Synchronization must take place shortly after an event has occurred. The more time elapses between the event and synchronization, the larger the error grows. The synchronization protocol must therefore have some kind of pull mechanism to trigger the execution of the synchronization process. RBS, TPSN, TSync and LTS support post-facto synchronization.

5.5 Absolute or relative time synchronization

The clocks in a WSN can be synchronized with respect to a global timescale or they can keep their own time but keep track of the time offsets relative to other nodes.

In the first case, all clocks will have the same time after synchronization. This is the most common way of synchronizing, used by TPSN, TDP, TSync and LTS. These protocols, except DTP, synchronize all nodes in the network to the clock of a master node. When a node is located further away from the master node, the synchronization error will be larger.

DTP works in a slightly different way, because the master nodes are re-elected every synchronization cycle. The clocks will therefore not be synchronized to a single master node's clock, but will converge to an equilibrium time.

RBS is a protocol that uses relative time synchronization. This means that the clocks of all nodes do not need to have the same value at the same time. The nodes will know their offset to the other nodes in the network. No master node is needed to which all nodes will synchronize.

When a WSN is connected to the outside world, it may be needed to synchronize to an external timescale. A global timescale that is commonly used is Universal Time Coordinated (UTC). This timescale is typically distributed via radio systems or GPS [EGE02].

Synchronization to external timescales is supported by all protocols. In protocols that use absolute time synchronization, this can be reached when the master node has access to the global timescale. In RBS, nodes can calculate their time offset to a node that has access to the global timescale.

5.6 Robustness

As said, WSNs are naturally very dynamic. Nodes may move, batteries get depleted or nodes may fail. This can have its effects on the time synchronization in the WSN. A robust protocol should still work when the network changes.

Protocols that use a static hierarchy are very sensitive to changes in the network. When a single node fails, all nodes that are below it in the hierarchy may get unsynchronized because they can no longer reach the master node. When the master node fails, the whole network can no longer be synchronized. This problem can occur in TPSN, TSync and LTS. Because TDP regularly changes the masters and creates a new hierarchy, it will much better adapt to network changes.

RBS is not based on a hierarchy, and is therefore probably the most robust protocol in this paper. The protocol is not sensitive to changes in the network. When the network topology changes, the conversion routes will also change, but as long as all nodes are connected, they can all still be synchronized.

5.7 Scalability

When a WSN is extended, the computing time used by the new nodes should automatically be synchronized to the time of the existing nodes in the network. When TPSN, TSync or LTS are used, the new nodes will first need to determine their position in the hierarchy. In RBS and TDP, the new nodes do not need to perform any special actions. They can just behave like the existing nodes that are already in the network.

When a WSN grows large, the synchronization error may increase. This is especially a problem in TPSN, TSync and LTS, where static master nodes are used. The time must be sent through the whole network, increasing the error at every hop.

As explained in section 3.2.2 this can result in a large synchronization error between nodes that are close to each other. A possible solution to this problem is to use multiple masters spread through the network. These masters then must be externally synchronized to each other, for example by using GPS receivers.

5.8 Hardware requirements

Some of the synchronization protocols have special hardware requirements. All protocols make use of the broadcasting ability of the nodes. Broadcasting is very common in wireless networks and necessary for detecting neighbor nodes.

The protocols that perform pair wise synchronization assume that the transmission time is symmetric. This means that the transmission time from node A to node B is the same as the transmission time from node B to node A.

The TSync protocol uses multi-channel radios to reduce packet collisions. Multi-channel radios are capable of communicating on more than one frequency channel. Sensor nodes equipped with these radios are becoming increasingly more common [DAI04].

6. CONCLUSIONS AND FUTURE WORK

Time synchronization is crucial in WSNs. WSNs are in many ways different from traditional wired networks, which makes existing time synchronization protocols as NTP inapplicable. In this paper five time synchronization protocols for WSNs are described. Further, they are compared using a number of criteria. This comparison is made only based on the available literature.

When we look at the accuracy of the protocols, we do not see very big differences. The average synchronization errors of the different protocols are approximately equal to each other. RBS and DTP support tuning, allowing to make a trade-off between accuracy and energy consumption.

To determine the energy consumption of the protocols, the number of messages is counted that must be sent for a single synchronization round. TSync seems to be the most efficient protocol at this point.

The robustness is another very important issue for time synchronization protocols. RBS and DTP are probably the most robust protocols, because they are not dependent on a static hierarchical structure in the network.

In extreme circumstances, the RBS protocol will probably be the best time synchronization protocol to use. It is much more robust and dynamic than protocols based on a hierarchy. DTP is a good alternative.

When the circumstances in which the WSN is deployed are less extreme and the network is less dynamic, the protocols that are based on a hierarchy will also perform well. Especially TSync can be more energy efficient than other protocols. These protocols work very similar to traditional time synchronization protocols, which can be an advantage when communication with wired networks is necessary.

This paper is purely a literature study. For future research, a good thing would be to do practical tests on these protocols. When the protocols are all tested in the same environment and under the same circumstances, the results will be clearer, especially for the comparison of accuracy and energy efficiency.

REFERENCES

[AKY02] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. *A survey on sensor networks*. IEEE Communications Magazine, 2002.

[BER00] H.G. Berns and R.J. Wilkes. *GPS Time Synchronization System for K2K*. IEEE Transactions on Nuclear Science, 2000.

[DAI04] H. Dai and R. Han. *TSync : A Lightweight Bidirectional Time Synchronization Service for Wireless Sensor Networks*. ACM Sigmobile Mobile Computing and Communications Review, Volume 8 Issue 1, January 2004.

[EGE02] J. Elson, L. Girod, D. Estrin. *Fine-grained network time synchronization using reference broadcasts*. ACM SIGOPS Operating Systems Review, Volume 36 Issue SI, December 2002.

[ELS01] J. Elson and D. Estrin. *Time Synchronization for Wireless Sensor Networks*. Proceedings of the 15th International Parallel and Distributed Processing Symposium., IEEE 2001.

[ELS02] J. Elson and K. Römer. *Wireless Sensor Networks: A New Regime for Time Synchronization*. In Proceedings of the First Workshop on Hot Topics In Networks (HotNets-I), Princeton, New Jersey, October 2002.

[ELS03] J. Elson. *Time Synchronization in Wireless Sensor Networks*. International Parallel and Distributed Processing Symposium, 2001.

[EST02] D. Estrin, L. Girod, G. Pottie and M. Srivastava. *Instrumenting the world with wireless sensor networks*. In International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2001), Salt Lake City, UT, May 2001.

[GAN03] S. Ganeriwal, R. Kumar and M.B. Srivastava. *Timing-sync protocol for sensor networks*. Proceedings of SenSys, 2003.

[GAY05] D. Gay, P. Levis and D. Culler. *Software Design Patterns for TinyOS*. To appear in Proceedings of the ACM SIGPLAN/SIGBED 2005 Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES'05), Chicago, June 2005.

[GOT02] T. Gotoh, K. Imamura and A. Kaneko. *Improvement of NTP time offset under the asymmetric network with double packets method*. Precision Electromagnetic Measurements, 2002. Conference Digest 2002 Conference on 16-21 June 2002. Pages 448-449.

[GRE03] J. van Greunen and J. Rabaey. *Time synch and localization: Lightweight time synchronization for sensor networks*. Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications, September 2003.

[GUR03] O. Gurewitz, I. Cidon and M. Sidi. *Network Time Synchronization Using Clock Offset Optimization.* Network Protocols, 2003. Proceedings. 11th IEEE International Conference on 4-7 Nov, 2003. Pages 212-221.

[HIL03] J. Hill. *System Architecture for Wireless Sensor Networks*. PhD thesis, UC Berkeley, May 2003.

[JUA02] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh and D. Rubenstein. *Energy-efficient Computing for Wildlife Tracking: Design Tradeoffs and Early*

Experiences with ZebraNet. In Proceedings of the 10th Intl Conference on Architectural Support for Programming Languages and Operating Systems, San Jose, CA, Oct 2002.

[MAI02] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler and J. Anderson. *Wireless sensor networks for habitat monitoring*. In ACM Workshop on Sensor Networks and Applications, 2002.

[MAR04] M. Maroti, B. Kusy, G. Simon, A. Ledeczi. *The Flooding Time Synchronization Protocol*. In Proc of The Second ACM Conference on Embedded Networked Sensor Systems (Sensys), November 2004.

[MEG01] S. Meguerdichian, F. Koushanfar, M. Potkonjak, M. B. Srivastava. *Coverage Problems in Wireless Ad-hoc Sensor Networks*. In Proc. IEEE Infocom 2001, Vol 3, pp. 1380-1387, Apr. 2001.

[MIL92] D. L. Mills. *Network Time Protocol (Version 3). RFC1305*. March 1992. [SU05] W. Su and I. F. Akyildiz. *Time-Diffusion Synchronization Protocol for Wireless Sensor Networks*. IEEE/ACM Transactions On Networking, vol. 13, no. 2, April 2005.

[MIL96] D.L. Mills. *Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI. RFC2030*. October 1996.

[TUL04] D. Tulone. *A resource-efficient time estimation for wireless sensor networks*. Proceedings of the 2004 joint workshop on Foundations of mobile computing, October 2004.

[PAL04] S. PalChaudhuri, A. K. Saha and D. B. Johnson. *Adaptive clock synchronization in sensor networks*. Proceedings of the third international symposium on Information processing in sensor networks, April 2004.

[PIN03] S. Ping. *Delay Measurement Time Synchronization for Wireless Sensor Networks*. Intel Research Berkeley Lab, June 2003.

[POT00] G.J. Pottie, W.J. Kaiser, L. Clare and H. Marcy. *Wireless integrated network sensors*. Communications of the ACM, 2000.

[ROM04] K. Römer and F. Mattern. *The design space of wireless sensor networks*. Wireless Communications, IEEE, Volume 11, Issue 6, Dec. 2004 Pages 54-61.

[SIC03] M. L. Sichitiu and C. Veerarittiphan. *Simple, Accurate Time Synchronization for Wireless Sensor Networks*. Proc. of the IEEE Wireless Communications and Networking Conference (WCNC), 2003.