Energy-efficient Sensing Data Delivery for Low Power Environmental Sensors

Deawoo Kim, Jinhwan Jung, Hankyeol Lee, and Yung Yi

Abstract Sensor networks are becoming extensively used to effectively and autonomously monitor our environment, where examples include environmental and habitat monitoring, structural health monitoring and condition-based equipment maintenance, and disaster management and emergency response. One of the popular and primary mechanisms for achieving low energy consumption in energy-constrained WSNs (Wireless Sensor networks) is duty cycling where each node periodically alternates the states of awake and dormant, motivated by the fact that a non-negligible portion of energy is consumed when in the idle listening state. In this chapter, under the framework of duty cycling, we survey the four key components for energy-efficient delivery of sensing data: (i) MAC (Medium Access Control), (ii) routing, (iii) wake-up scheduling, and (iv) time synchronization. These four components are often coupled in many cases, where they have to be collaboratively optimized for better energy efficiency in operating WSNs. We survey the recent advances in those four components and conclude with the discussion of the future directions in this area.

Key words: Wireless Sensor Network, Duty Cycle, Wake-up Scheduling, Time Synchronization, Sensor Network Routing, MAC

Deawoo Kim
Electrical Engineering, KAIST, e-mail: dwkim@lanada.kaist.ac.kr

Jinhwan Jung
Electrical Engineering, KAIST, e-mail: jhjung@lanada.kaist.ac.kr

Hankyeol Lee
Electrical Engineering, KAIST, e-mail: mrrays@kaist.ac.kr

Yung Yi
Electrical Engineering, KAIST, e-mail: yiyung@kaist.edu

1 Introduction

Environmental Sensor Networks: Examples and Architecture

Sensors and their formation of "networks," often referred to as wireless sensor networks (WSN), are being increasingly used for environment monitoring, becoming an integral part of our lives, e.g., habitat monitoring, structural health monitoring and condition-based equipment maintenance, and disaster management. Exemplary projects include monitoring of sea bird nesting environment [21] and life of tall redwood trees [34], an agricultural monitoring system [11], a vineyard monitoring system [3], and habitat monitoring system [6]. Such trends have continued to date, and recent, extensive efforts in the research and commercialization domains have enabled the actual implementation and deployment of such environmental sensor networks, often tailored to the unique requirements of each of sensing and monitoring applications.

A WSN is naturally the overarching mixture of technologies from a variety of different areas: sensing, communication, computing, where the main task of a sensor node is to detect events, perform local data processing, and then transmit/receive processed sensing data. Fig. 1 shows a typical architecture of a WSN.

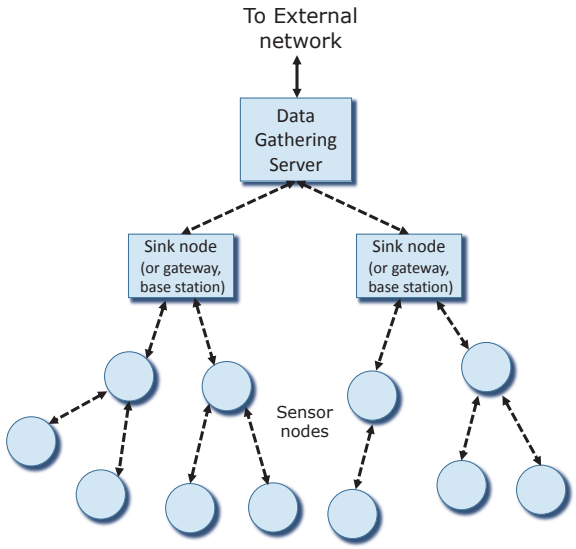

Fig. 1 General Architecture of Wireless Sensor Networks

As depicted in Fig. 1, sensor nodes are distributed in some geographical locations in a random or controlled manner, being responsible for sensing and gathering data. Sensors form a single or multiple wireless multi-hop networks, requiring to be operated in an energy-efficient manner, e.g., via duty cycling (see Section 2 for more details). Some sensor nodes are connected to sink nodes (also called base stations or

gateways), which in turn are connected to a data gathering server for storing sensed data and offering the data to other processing units for further usage including decision making through data reprocessing, graphical visualization, etc. We sometimes refer to just the collection of sensor nodes before the sink nodes as a WSN, because networking there is one of the most challenging parts in WSNs due to (a) limited capability of energy and (b) harsh environments (e.g., scalability and deployment in the field with restricted human controllability).

Challenges of WSN

Developing "efficient" communication protocols in all layers for sensing data delivery in WSNs possesses significant challenges, compared to those in other wireless multi-hop networks. Thus, a suitable balance between performance (e.g., throughput and latency) and energy-efficiency should be stroke when designed, which in turn depends on the target application scenarios and the limiting budget in terms of energy.

(a) *Limited energy budget:* Sensor nodes are typically battery-powered and tend to last only during a limited amount of time. Thus, different from "traditional" wireless networks, which mainly target at improving conventional performance metrics, e.g., throughput and delay, the key focus of WSN protocols lies in low power consumption. External power sources through energy harvesting (e.g., solar cells) are often attached to sensor nodes, but they tend to be irregular, necessitating more careful and smarter energy management skills, adding extra complexity to communication protocols. The challenges in networking protocols for good energy maintenance are added by sensor nodes' local processing. For example, they do not deliver the raw sensed data to a sink node, but a certain degree of data fusion may occur, for which sensor nodes use their processing capabilities to locally carry out simple computations and transmit only the required and partially processed data.

(b) *Domain/application dependence:* Applications and domains where WSNs are used are highly diverse. For example, in structure health monitoring[16], sensor nodes are statically deployed in, e.g., walls or bridges, whereas in monitoring the life of zebra in wild areas [40] sensors are highly mobile, where inter-sensor communications are significantly opportunistic. In many sensing applications of monitoring physical phenomena, sensor nodes can acquire LoS (Line-of-Sight) path for their communications, whereas they are placed at the environments that have many blocking materials. These diverse applications and domains require WSN protocols to be developed and applied in dependence on where and when they are installed and deployed.

(c) *Scalability, Correlation, and Self-X:* The position of sensor nodes may be often incontrollable, leading to a random deployment, in which case it is possible to have both densely and sparsely deployed areas. Especially, in the densely-deployed area, scalability can be a major issue for data delivery, and in particular, since sink nodes are the points of data aggregation, and thus highly asym-

metric in data transmission and reception, there exists an issue of single point of failure. Due to the short transmission ranges of sensors, a large number of sensor nodes should be densely deployed for connectivity and neighboring nodes may be very close to each other. Hence, multi-hop communication is exploited in communications between nodes since it leads to less power consumption than the traditional single hop communication. Also, low-cost sensor nodes often fail to operate due to bad weathers or the action of wild animals. Such frequent, unexpected faults and scalability issues need WSN protocols to be designed in such a way that they are self-X capable (e.g., self-healing and self-organizing). Furthermore, the dense deployment coupled with the physical properties of the sensed phenomenon may introduce correlation in spatial and temporal domains, e.g., multiple sensors sense similar features. As a result, the spatio-temporal correlation-aware protocols may be necessary for better energy-efficiency and smaller data intensity.

Who Is the Energy Killer?

As previously mentioned, WSN incorporates the technologies of sensing, communication, and computing. Thus, the amount of power consumption can hence be studied, being divided into three parts—sensing, communication, and data processing—each performed by sensors, CPUs/MCUs, and radios, respectively. A breakdown of the power consumption of a MicaZ sensor node is shown in Figure 2 from [25]. This shows that a sensor node expends maximum energy for data communication, implying the paramount importance of energy-efficient data communication protocols in WSNs.

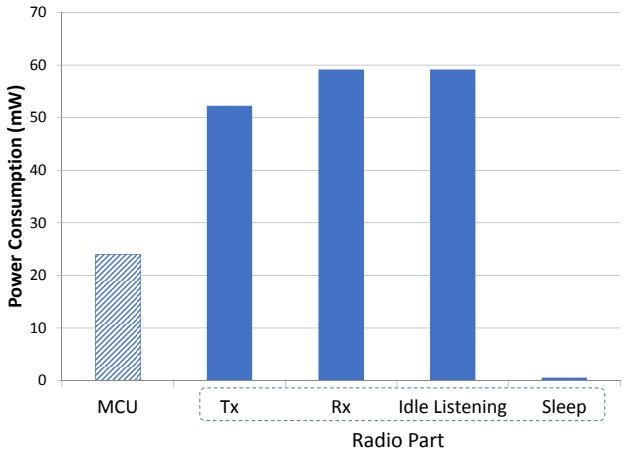

Fig. 2 Power Consumption Breakdown of MicaZ. Source: [25]

In this chapter, we summarize some of the existing proposals of energy-efficient WSN networking protocols by classifying them into four major components: (i) MAC (Medium Access Control), (ii) routing, (iii) wake-up scheduling, and (iv) time synchronization for which we provide a summary of issues of those components.

MAC (Medium Access Control)

One of the primary mechanisms for achieving low energy consumption is duty cycling, where each sensor node periodically cycles between awake and dormant states. The key parameters in characterizing a rule of duty cycling include the durations of sleep and dormant states, which actually depends on the application traffic pattern and intensity. Given duty-cycled sensor nodes, the challenges faced by MAC designers are how to control medium access in an energy efficient manner. In particular, when a node is in a dormant state, coordinating the communication between a sender and its intended receiver is hard.

Routing

Routing is another central component in WSNs, because, as stated earlier, sensor nodes' communication range is relatively small, and thus inevitably in need of multi-hop communications. Routing becomes challenging in WSNs because of energy efficiency, scalability, addressing, robustness, and application-dependency. Especially, different target applications, which are either time/event-driven or query-driven, should result in totally different routing protocols for good performance.

Wake-up scheduling

Although energy-efficiency is one of the primary concerns in WSNs, time-urgency often becomes important in some applications, e.g., fire alarm. In particular, in duty-cycled WSNs, unless data is delivered over many hops under one wakeup duration, it would take much time for data to arrive at a sink node. In other words, a sender has to hold the transmission and wait until the receiver wakes up based on its schedule. Note that MAC is just responsible for a single-hop communication with focus on energy efficiency, being unable to offer the solutions for small multi-hop latency. Thus, additional control mechanism should be involved, one of which is to appropriately schedule wake-up times of nodes. One can propose a "pipeline mechanism" to ensure that each relaying of a packet can catch up the wakeup time slot of the forwarder.

Time synchronization

In distributed systems, a node is typically equipped with its internal clock. A WSN is not the exception, used for various operations in sensing, processing, and communication. However, the clock in a low-cost sensor node is often a low-quality one, losing its accuracy when not often synchronized and being too sensitive to external

conditions such as heat and humidity. In particular, communication inherently requires a collaborative task between senders and receivers, and thus many protocols in MAC and routing sometimes require time synchronization among nodes. For example, there exist a variety of MAC protocols based on tight sensor synchronization, e.g., [37, 7].

2 MAC (Medium Access Control)

2.1 Design Guidelines and Challenges

Challenges

A decade of technological advances in hardware and software have witnessed a stiff improvements in one-chip, light-weight sensor nodes with low-cost. A sensor node is composed of memory, MCU, several sensors. However, those one-chip solutions significantly limit the communication coverage, inevitably resulting in a large number of multi-hop communications to deliver intended data to a sink as well as a large number of sensor nodes need to be scattered and installed for successful environmental monitoring. These unique features of WSNs are in stark contrast to those in other wireless multi-hop networks, generating the distinct challenges in the design of MAC protocols. Of primary concern is the limited power source in wireless sensor networks, and thus in addition to a pure function of MAC, i.e., how to share the medium, a new role of how to save power consumption is added to sensor nodes, so as to prolong the lifetime of network. This energy efficiency becomes much more important in the scenario where sensors are not fully controllable—once they are installed, their batteries are unable to be recharged.

Where is energy used inside MAC?

There are many root causes of energy waste, as summarized in what follows. MAC protocols should be aware of these energy waste sources, for which they should be intelligently designed for high energy efficiency.

- *Collision:* Despite MAC's major role of scheduling link transmissions in a distributed manner, nothing is perfect in the sense that it is possible that a node may receive multiple packets simultaneously (mainly due to MAC's imperfectness), in which case collision occurs. Collisions lead to unnecessary transmissions, which further generate retransmissions, consuming a certain amount of energy.
- *Overhearing:* Wireless is basically a broadcast medium. Thus, when a packet is generated for a specific receiver, the packet is first broadcast to the neighbors each of which decodes the packet's destination address to check out whether it is intended for itself or not. Thus, overhearing transmissions in the neighborhood is inevitable, which is also the source of energy waste.

- *Idle listening:* Recall the breakdown of power consumption of a MicaZ sensor node in Fig. 2, demonstrating that the power consumed in idle listening is non-negligible. Idle listening is naturally performed very often, when a receiver (or even a transmitter) should listen to an idle channel in order to detect the incoming transmission. As an example, CSMA (Carrier Sense Multiple Access), which is one of the most popular wireless MACs, largely relies on this idle listening, because its core feature is that whenever nobody accesses the medium, I access it.
- *Control overhead for communication coordination:* As mentioned earlier, the unique function of MAC protocols in WSNs lies in saving energy as much as possible. As will be detailed shortly, one of the popular mechanisms is duty cycling, i.e., nodes go to the sleep state, escaping from idle listening, and wake up for the activity of packet transmission and reception. However, depending on how nodes are synchronized, it is often the case when a transmitter and a receiver should wait for each other to set up an appropriate communication, for which a certain procedure of signaling is needed. Well-designed communication coordination can reduce the number of control messages and the time to wait for setting up communication. In this way, we can considerably save energy by avoiding idle listening and overhearing.

2.2 Taxonomy: MAC based on Duty Cycling

2.2.1 Duty cycling

What would be a natural way of saving power consumption (generated by communication) of sensor nodes? Everybody would agree that it is to turn off the transceiver circuitry whenever possible. In WSNs, that is typically accomplished by so-called *duty cycling*. In a "basic" duty cycling based MAC, each sensor node is programmed with a duty cycle a priori, and it alternates between wake-up and sleep states following the programmed duty cycle. A node wakes up periodically to transmit or receive packets from other nodes. Usually after a node wakes up, it listens to the channel for any activity before transmitting or receiving packets. If no packet is to be transmitted or received, the node returns to the sleep state. In the rest of this chapter, we use the term 'sleep period' and 'wake-up period' to mean the times when nodes turn off and turn on their transceivers, respectively, and the concatenation of a sleep and wake-up periods is referred to as a 'working period' (or simply 'period' unless confusion arises). Then, duty cycle is measured as the ratio of wake-up period to the working period which gives an indicator of how long a node spends in the wake-up period. When we mention "low duty cycle", it means that the duration of being in the sleep state is much longer than that of being in the wake-up state. Duty cycle should be determined depending on the target applications, i.e., it differs depending on the amount of sensed traffic generated. Whenever traffic is generated in an

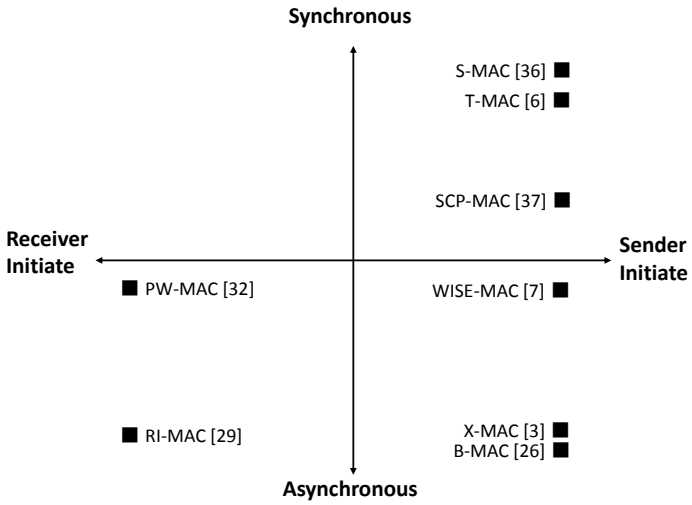

Fig. 3 Taxonomy of some WSN MAC Protocols: Asynchronous vs. Synchronous, and Sender-initiated vs. Receiver-initiated

event-driven manner and thus the amount of traffic generated becomes situation-dependent, it is often necessary to adaptively change the duty cycle in a distributed manner. The ideal case that such duty-cycle based MAC works perfectly is when a transmitter and its intended receiver wake up at the same time and spends their energy purely for communication and go back to sleep. However, such an ideal case is very challenging to achieve, because (i) heavy synchronization overhead is required to synchronize the nodes, or (ii) being without synchronization inevitably leads to large clock drift, and thus the underlying MAC should take such drift into consideration at the cost of extra energy consumption.

2.2.2 Taxonomy

There are an extensive array of sensor MAC protocols based on duty cycling. Clearly, it is impossible to survey such all the exhaustive list. In this survey article, we apply a certain classification criterion to them, and summarize a partial list of them, which, we think is enough to present diverse possible design angles. Our criterion of classifying the existing MAC protocols is two-fold: Asynchronous vs. synchronous, and sender-initiated vs. receiver-initiated. These two criteria tell us (i) *how* to coordinate communication and (ii) *who* initiates communication. Fig. 3 shows some of sensor MAC protocols, according to our classification criteria.

How to Coordinate: Synchronous vs. Asynchronous

- *Synchronous:* All the nodes in a "neighborhood" have the same time of wake-up period, for which they run a certain synchronization procedure with help of message exchanges or GPS (see Section 3.3 for more details).
- *Asynchronous:* Sensor nodes' wake-up period is not synchronized. Thus, whenever a transmitter has a packet to deliver, it detects when its intended receiver is ready for reception and carries out packet delivery.

Note that in this article we will exclude the case when GPS is used in synchronization, because GPS is known to consume a large amount of power, compared to other components, and thus it is rarely used in MAC of WSNs. As shown in Fig. 3, a protocol may not be perfectly synchronous or asynchronous. This means that a protocol may be designed to infrequently synchronize the nodes, where the protocol is *partially synchronous.*

Again, as mentioned earlier, synchronous vs. asynchronous deals with the issue of how to coordinate the communication between a pair of nodes. Synchronization frequency takes a tradeoff point between the synchronization overhead and the control overhead due to resulting clock drift; More frequent synchronization leads to smaller clock drift, yet generating larger synchronization overhead, whereas less frequent synchronization results in larger clock drift, and thus more control overhead to detect when the receiver is ready for packet reception is incurred. What is the best trade-off relies on the underlying traffic generation intensity as well as how smartly synchronization mechanism and control signaling to tackle clock drift are designed in a protocol.

Who Initiates: Sender vs. Receiver

- *Sender-initiated (SI):* In this approach, a sender first expresses its intention of communication via request signal (e.g., preamble, control or even data packet themselves) over a time interval during which its receiver detects whether somebody wants to communicate with itself.
- *Receiver-initiated (RI):* Different from the sender-initiated approach, in this approach, a node expresses its readiness to receive packets to neighboring potential senders.

It is natural that more energy is consumed at senders and receivers in SI and RI approaches, respectively. Thus, energy efficiency for both approaches becomes different depending on the traffic generation conditions. In SI approach, since energy is used whenever packets are backlogged at a sender, in the case when traffic is small, it is energy efficient. On the other hand, in RI approach, a receiver's signal is not generated by packet generations, and thus it becomes more energy efficient when there are more packets to communicate in the network. Note that in an "ideally" synchronous protocol that perfectly synchronizes nodes (thus very high synchronization frequency), the issue of who initiates communication is of little interest, because both the nodes know exactly when to communication. However, as

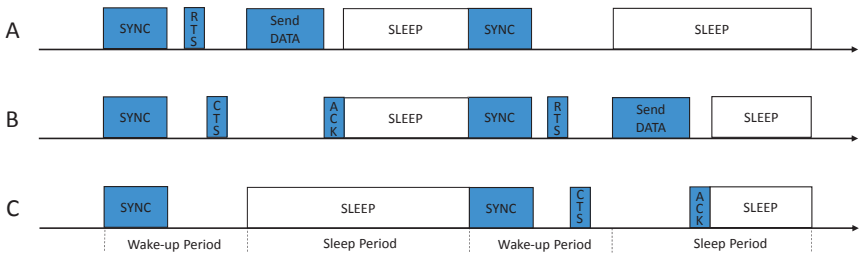

Fig. 4 S-MAC Protocol

mentioned earlier, it is very difficult to achieve perfect synchronization, and also, depending on synchronization frequency, clock drift is inevitable. We finally comment that what is more efficient, SI or RI approach, also depends on the amount of energy consumption in reception or transmission, which also relies on the hardware architecture employed to configure a sensor node. For example, in Mica2 nodes [24], transmission energy is higher than receiving energy, but the opposite is true for MicaZ nodes [25]. In the rest of this section, we summarize the example MAC protocols in each of four categories, which are well-known in the research community.

2.3 S-MAC [37] and T-MAC [7]

Basic Mechanism

In S-MAC, a fixed time interval, called a frame, is divided into sleep and wake-up periods, also referred to as *periodic listen and sleep,* where each node tries to synchronize with its neighboring nodes by performing some synchronization-related signaling protocol. Through this signaling protocol, a subset of nodes form a virtual cluster, meaning the nodes inside the cluster are synchronized and thus share the wake-up time. The wake-up period is divided into two sub-phases: SYNC and RTS/CTS signaling. In each of these two phases, each node accesses the medium to transmit and receive SYNC, RTS, and CTS messages based on the famous CSMA with random back-off. See the example in Fig. 4, where node A intends to transmit data to node B. After SYNC phase, A sends RTS and receives CTS from B, the actual DATA is transmitted followed by ACK from B, after which both nodes go to sleep until the next SYNC phase. Note that the time till the next SYNC phase is shared by the nodes inside each cluster, i.e., local synchronization via sleep schedule sharing mechanism. This local synchronization and the notion of virtual cluster permit some nodes to be the "border" nodes, and thus be synchronized with multiple clocks. These border nodes need to wake up more often than other normal nodes, but they are claimed to be quite rare.

Advanced Features

Overhearing avoidance. It is possible that a node, who is not an intended receiver, listens to the channel and overhears RTS or CTS packets, just like the node C in Fig. 4. The node C is able to detect that it is not an intended receiver by extracting RTS or CTS packet's destination MAC address. In this case, S-MAC allows such nodes to directly go to sleep during nodes A and B's transmission time to reduce energy consumption. The transmission time can be obtained from the NAV (Network Allocation Vector) field similarly to 802.11-based WLAN.

Multi-hop awareness: Adaptive Listening. One of the pitfalls of the S-MAC's basic mechanism is that in one working period, only one-hop packet delivery is possible, which results in a large latency whenever a packet needs to be relayed to a target sink node over a multi-hop path. S-MAC adds a feature of *adaptive listening* to reduce such multi-hop latency. Adaptive listening is a mechanism that nodes with NAV information wake up around the time when data transmission is expected to be finished and the nodes wait for a short time listening for any incoming packets. This adaptive listening is a mechanism that is not coupled with the routing layer and operates without any knowledge of route information and any information about scheduling of neighbor nodes. Therefore, this scheme may not always decrease the latency. For example, if the receiver lies in the border of a virtual cluster, adaptive listening cannot wake up the nodes in other cluster. In this case, without any benefit of latency, energy consumption may increase due to the adaptive listening of all the neighbor nodes that overhear a transmisstion. Thus adaptive listening is regarded as a best-effort solution that makes S-MAC multihop-friendly.

T-MAC: Extension of S-MAC

S-MAC is improved by T-MAC that employs a timeout based adaptive listening to make it dynamically adaptive to the amount of traffic. In T-MAC, a wake-up period ends whenever sensing tells us that the medium is idle during the duration of the pre-specified timeout. Thus, a node does not need to remain idle for the remaining duration of the wake-up period after SYNC, when there is no activity in the network.

Pros and Cons

S-MAC has been proposed as a first-generation MAC protocol in WSNs, probably one of the "most synchronous" protocol in the sense that in every working period, it tries to (locally) synchronize inside a group of nodes, called a virtual cluster. This synchronous protocol such as S-MAC is not energy efficient especially when traffic load becomes low, because energy consumption through synchronization overhead dominates even in absence of traffic. T-MAC improves S-MAC by adaptively adjusting sleeping period through a notion of timeout.

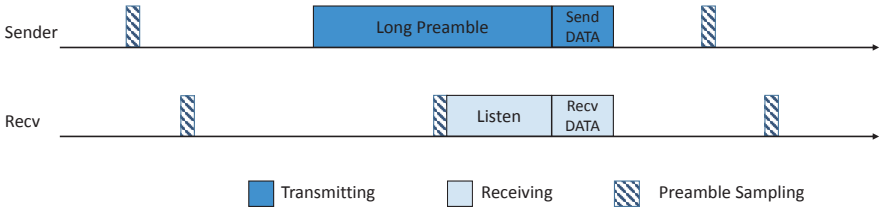

Fig. 5 B-MAC Protocol

2.4 B-MAC [27] and X-MAC [4]

Basic Mechanism

B-MAC significantly differs from S-MAC in that it works without any synchronization effort, mainly motivated by the fact that a synchronous protocol such as S-MAC has low energy efficiency, in particular when the offered traffic load is low. Thus, B-MAC removes the heavy synchronization overhead, and "contacts" an intended receiver, if need be, on a on-demand basis. Then, the natural question is how the sender contacts its receiver? B-MAC carries out the receiver contact using a long preamble signal. In other words, the sender, prior to sending a data packet, first transmits a long preamble signal which lasts longer than the receivers sleep period, ensuring that the receiver wakes up at least once during the preamble length, so that it can detect the sender's intention of contact. The receiver has an advanced, energy-efficient mechanism of sensing busy activity in the medium (see the details later). If the medium is sensed busy, the receiver turns on its transceiver until the data packet is received or timeout occurs.

LPL and CCA. The receiver's preamble sampling (i.e., the task of detecting a transmitter preamble for its intended transmission) is improved for better energy efficiency and accuracy than that in other wireless networks, e.g., CSMA in 802.11. That is done by two mechanisms, called LPL (Low power listening) and CCA (Clear Channel Assessment). The main idea behind LPL is to send a preamble before each packet to wake up the intended receiver with the goal of minimizing the listen energy cost associated with the fixed duty cycle protocols. Accordingly, each node periodically wakes up, turns on its radio, and checks for activity on the channel. During this small period, if activity is detected on the channel, the node stays in the receive mode. However, if no activity is detected, then the node switches back to sleep state. Towards successful LPL operation, a mechanism of detecting channel activity that is energy-efficient and exact is necessary, otherwise collisions or medium under-utilization is generated.

The main goal of the CCA mechanism is to differentiate between noise and a signal to accurately assess the channel activity through a software approach, where a multiple of channel measurements are performed, motivated by the fact that frequent signal fluctuations in the received signal strength of a receiver are observed, even during a lack of packet transmissions.

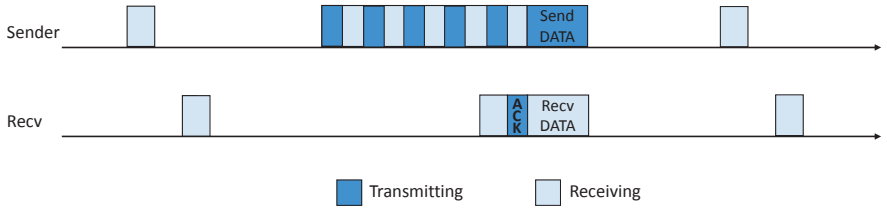

Fig. 6 X-MAC Protocol

Extension: X-MAC

A problem of B-MAC in terms of energy efficiency and latency lies in the fact that a pair of sender and receiver should wait until the long preamble ends in order to start communication. The natural extension should be the one which removes such unnecessary waiting time, motivating the development of X-MAC. To that end, X-MAC uses a *strobed preamble* that consists of a sequence of spaced short preambles prior to data transmission, as depicted in Fig 6. In this case, if the receiver samples such a short preamble piece, it can notify the sender of its sampling result, so that they can immediately start communication. To make it possible, the target receiver address is included in each short preamble, which also helps unintended receivers to go to sleep immediately and also allows only the intended receiver to send an early ACK to the sender. X-MAC provides more energy efficient and lower latency performance operation by reducing both preamble length at transmitters and idle listening at receivers.

Pros and Cons

Compared to synchronous protocols such as S-MAC, a significant amount of idle listening can be reduced, especially when the offered traffic is low, thus more energy-efficient under the low duty cycle scenarios. For the case of high offered load, B-MAC's LPL requires a long preamble to be sent, and thus a lot of performance inefficiency is expected. Also, long preamble based scheme such as B-MAC leaves a lot of room for performance improvement, when the duty cycle is extremely low, because a very long preamble should be sent, which also holds for X-MAC in the worst case.

2.5 SCP-MAC [38]

S-MAC and B-MAC can be regarded as two extreme protocols in terms of the frequency of synchronization, where two different kinds of energy waste sources are observed. In S-MAC, nodes are locally synchronized every working period, where synchronization overhead is the major source of energy waste, whereas in B-MAC nodes are completely unaware of synchronization (thus leading to clock drift of the

length of working period), where the overheads for communication coordination through preamble sampling are the major source of energy waste.

SCP (Scheduled-Channel-Polling) is a hybrid protocol that hopefully combines the advantages of those two extreme protocols, i.e., combines LPL with a certain degree of synchronization. Thus, SCP-MAC can be positioned by its feature of "infrequent synchronization and B-MAC-like protocol for combating against small clock drift". Fig. 7 exemplifies the protocol behavior of SCP-MAC, where whenever a sender has data to transmit, it transmits a tone signal (which corresponds to a long preamble in B-MAC, which, however, should be long due to no existence of synchronization), sampled by the receiver.

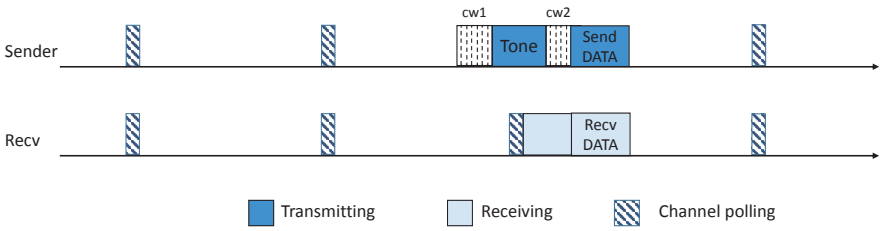

Fig. 7 SCP-MAC Protocol

To elaborate how SCP works, see Fig. 7. All nodes in the network wake up at a regular interval and perform a synchronized carrier sense. In case when multiple senders contend around a receiver, the receiver can waste energy by failing to communicate to one sender. Thus, a sender first contends for the channel before the scheduled wake-up (cw1 in Fig. 7). In polling the channel, receivers should consider the possible clock drift due to infrequent synchronization, requiring a prolonged preamble, referred to as *tone* in Fig. 7. However, instead of a long contention resolution phase as in the conventional contention-based MACs, SCP-MAC employs a two-phase contention resolution; SCP-MAC employs a second contention window after the wake-up time (cw2 in Fig. 7) to handle the case when multiple senders choose a same time slot, enabling to limit the length of cw1.

2.6 RI-MAC [30]

Basic Mechanism

RI-MAC is a representative protocol that is asynchronous and receiver-initiated. In asynchronous SI approaches such as B-MAC and X-MAC, a sender occupies the medium for a long time by sending a preamble signal to contact its receiver, and the data transmission occurs between those two after the receiver is detected by the long preamble signal. While the medium is occupied, clearly other nodes cannot access the channel, and thus when the traffic is bursty and is in the high load, low

throughput and large latency is unavoidable. On the other hand, in the receiver-initiated approach such as RI-MAC, rather than sending a long preamble such as in SI-based MACs, whenever a sender has data to transmit, it wakes up and waits silently for an intended receiver to wake up and notify the sender of its readiness, avoiding unnecessary channel occupation.

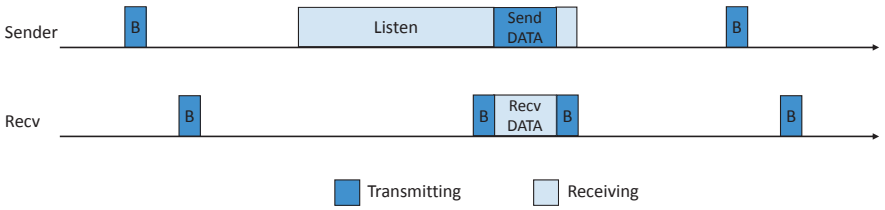

Fig. 8 RI-MAC Protocol

Fig. 8 depicts how RI-MAC operates. A receiver wakes up based on to its wake-up schedule (that is not synchronized with its neighbors), and broadcasts a beacon signal if the channel is clear. This beacon signal notifies the receiver's readiness to receive a packet to its potential senders. After transmitting a beacon signal, it waits for a data packet for a short time, and if there is no intention from a sender, it falls asleep again. A sender with pending data packets wakes up and stays active silently until the receiver broadcasts its readiness beacon signal. As soon as the sender receives the beacon signal from its intended receiver, it transmits the data to the receiver, which is acknowledged by an ACK beacon. This ACK beacon has two roles. First, it acknowledges whether the data packet is transferred successfully or not. If the data packet is transferred successfully, the sender can go to sleep state. However, the multiple senders' data transmission can lead to collisions, in which case the ACK beacon acts as a collision notification, and thus the senders randomly back off for retransmission. The second role of the ACK beacon lies in making contacts with other potential senders, where no data packet from the potential senders allows the receiver to move on to the sleep mode.

Pros and Cons

Compared to asynchronous SI approaches such as B-MAC and X-MAC, RI-MAC allows short medium occupation time for contact between a sender/receiver pair, leading to the improvement of throughput and latency. Also, it reduces energy consumption for bursty and high traffic loads. However, in traffic loads with low intensity (and thus with low duty cycles), it may consume more energy than SI MACs, because transmitting a beacon signal and waiting for data packets consume more energy than the LPL-based mechanism in asynchronous SI approaches.

2.7 Other MACs

Wise-MAC [8]

WiseMAC is an *asynchronous* and *sender-initiated* MAC protocol, working similarly to B-MAC. The long preamble signal in B-MAC causes energy-inefficiency and occupies the channel unnecessarily. In WiseMAC, a sender reduces the length of the preamble signal by scheduling wake-up times of its neighbors. Although the nodes are not completely synchronized and wake up in an asynchronous manner, each node learns the wakeup schedules of their neighbors. Once a node receives a DATA packet, it sends an ACK packet which is also used to inform its neighbors of the remaining time until the next wake up. In this way, a node is able to maintain a table of wake up schedule of all its neighbors, which enables a node to send packets with just a short preamble signal. Just the length of the preamble signal is enough as long as the signal covers the potential clock drift between the sender/receiver pair, turning out to be far shorter than that of B-MAC. This reduced preamble signal allows not only to save the energy of the sender node, but also reduce the overhearing of receiver nodes. Furthermore, the channel can be used more efficiently, because the time occupied by the preamble signal is also significantly reduced.

PW-MAC [33]

PW-MAC is an *asynchronous* and *receiver-initiated* MAC protocol. The key idea, similarly to WiseMAC, is that in PW-MAC wake-up time of the intended receiver is predicted in the following way: Every node is scheduled to wake up based on its own pseudo-random schedule generator, and with the given seed a sender is able to accurately calculate the wake up time of intended receiver. Note that a sender is initially unaware of the seed of its target receiver, and thus the sender stays in the active waiting for the beacon signal from the receiver, similarly to RI-MAC. However, once a packet is received by the receiver, the receiver transfers its pseudo random seed in its beacon signal, after which the wake-up can be predicted. In PW-MAC, similarly to WiseMAC, the sender is required to wake up a little earlier than predicted wake up time to appropriately cope with possible clock drift. Using this pseudo random number based wake-up scheduling, PW-MAC is capable of avoiding the consistent collisions among nodes that have similar wake-up schedule. However, calculating wake-up time prediction and storing neighbors' schedule are needed, leading to computational and spatial overhead in operating PW-MAC.

3 Routing, Wake-up Scheduling, and Time Synchronization

In this chapter, we discuss other components than MAC—routing, wake-up scheduling, and time synchronization—that are also critical for efficient delivery of data in WSNs. Due to space limitation, we focus on the key design challenges there and the

rough design angles in each of the components based on some of the representative protocols. We refer the readers to other nice surveys, e.g., [1, 31] and references therein for more details.

3.1 Routing

Challenges for Routing

We start by the component that is responsible for forwarding data from sensors to sink nodes, that is *routing*. Routing in WSNs resembles that in conventional wireless multi-hop networks in some sense, but there exist unique challenges that cannot be ignored as summarized in what follows:

- *Energy Efficiency:* As consistently highlighted throughout this chapter, the key factor in designing WSN network protocols is energy efficiency, and the routing is not an exception. A typical data transport path in WSNs is from a sensor to a sink, where routing contributes a lot to energy-efficient delivery of sensing data over such multi-paths. Clearly, different design angles possess different energy-efficiency strategies, necessitating a totally different twist of the routing protocols in normal wireless multi-hop networks. Routing protocols in WSNs should sometimes be jointly designed and optimized with other layers for maximum energy-efficiency.
- *Scalability/Robustness:* The number of sensors often reaches a large scale, up to an order of thousands, and the deployed node density can be highly heterogeneous, ranging from very sparse to very dense. Once a designed routing protocol needs heavy message passing, it becomes a disaster. A critical requirement of good routing protocols is a restricted number of control message exchanges, operating based only on local information. Furthermore, nodes are distributed in a variety of environments which are often tough and harsh, where node failures are very common. Routing protocols should provide strong robustness to work well even in such bad situations.
- *Addressing/Domain:* Allocating unique addresses to sensor nodes as in conventional networking systems may be inappropriate in large-scale WSNs. Strictly speaking, it is possible, but inefficient, because every packet should include the addresses of source/destination pair, taking up the large space in the packets' header. Fortunately, in WSNs, not a small number of applications do not necessarily necessitate such address-based identification. This implies that routing protocols and their addressing schemes should be customized by the target applications. For example, query-type applications typically require users to send a query that suffices to be supported by an attribute-based naming. Also, routing protocols should rely on the generated traffic pattern. For example, environment monitoring applications typically need static and robust routes to collect the peri-

odic status reports from the whole network. In contrast, event driven applications generate bursty and abrupt data, and commonly require in-time data delivery.

In this chapter, we choose some example routing protocols which we classify into (i) data-centric routing, and (ii) non data-centric routing. Data-centric routing is used for large-scale networks where it is hard to assign a unique ID to each sensor node. Hence, routing is performed based on queries requesting specific data. In non data-centric routing, each node has a unique ID, being similar to typical address-based routingg protocols. As we see, this classification is based on the fact that routing protocols in sensor networks tend to possess highly different features, depending on what type of application traffic is delivered. We comment that here by "simple routing", we mean the routing protocols where nodes simply entirely or selectively flood incoming packets.

3.1.1 Data-centric Routing

Flooding and Gossiping (Random Walk)

Flooding. One of the simplest routing mechanisms is to flood the incoming packets to my neighbors. Flooding is used in many types of computer networks, not necessarily in sensor networks. However, due to its extreme simplicity, its advantage is more envisioned in sensor networks where complicated processing may not be significantly desirable. However, as pointed out in literature, the weakness of flooding is (i) implosion of in-transit packets inside the network and (ii) energy-agnosticity that energy-efficiency is not explicitly considered in routing.

Gossiping. Another simple routing mechanism is to *selectively* flood the packets. Especially, we can consider a routing where a sensor node randomly selects a neighbor node for the next-hop transmission, which actually corresponds to a random walk over the nodes, also called *gossiping*. Clearly, this gossiping can significantly reduce the number of in-transit packets and save sensor nodes' energy, but at the cost of larger latency than flooding.

SPIN (Sensor Protocols for Information via Negotiation) [14]

SPIN is the first work of "non-trivial" data-centric routing protocol. The basic idea of SPIN lies in the reduction of data packets through a notion of *negotiation* that enables only interested sensor nodes to receive the intended sensing data. The negotiation is carried out through three messages: (i) ADV, (ii) REQ, and (iii) DATA. A node broadcasts ADV to its neighbors to transmit its intention of DATA delivery with the ADV containing a description of the intended DATA. Then, a set of neighbors which is interested in the received ADV responds by sending REQs, after which DATA is transmitted to the interested neighbors. SPIN protocol is refined from flooding so that data are routed based only on nodes' interest.

Directed Diffusion [15]

One of the popular applications in environmental sensor networks is that users query the monitoring status to the network, where a sink node broadcasts a query message (e.g., "what are the areas with humidity higher than some threshold?") into the network, to which sensor nodes respond by sending a reply packet. This query-type application does not necessarily require address-based routing, because the querier is not interested in communicating with some specific nodes, and thus so-called necessitating attribute-based naming.

Directed diffusion is designed to meet such a requirement, consisting of four phases: (i) the interest propagation phase that a sink node broadcasts its interest to the entire nodes in the network, (ii) the gradient phase that nodes' interest replies are collected to the sink nodes, (iii) reinforcement phase that the sink can select a particular path considering link quality and delay, and finally (iv) data delivery phase that a packet from a source to the sink is actually delivered.

3.1.2 Non Data-centric Routing

As opposed to data-centric routing, the class of routing protocols belonging to this category is similar to address-based routing, where whenever communication is needed, a node designates its receiver as a destination. The major interest here lies in how to route the packets in a scalable manner, so that data is guaranteed to be delivered with less energy and lower latency. Two families are notable, including cluster-based schemes and geographic schemes, which will be briefly summarized shortly. We end this section by presenting the routing protocol, referred to as RPL (Routing for low-Power and Lossy networks) [35], which is standardized in IETF.

Cluster-based Routing

This collection of protocols aims at minimizing energy consumption by performing a cluster-based operation. The goal is to dynamically select sensor nodes as cluster heads and form clusters in the network, and the communications inside the clusters are directed to the cluster head, which performs aggregation and is responsible for routing the packets to other cluster heads towards the sink, or directly communicates with the sink. Often, the cluster heads are dynamically changed in order to respond to the changes in the network, but more importantly to balance the nodes' energy consumption and thus maximize the network lifetime. Cluster-based routing protocols include LEACH [13], PEGASIS [32], TEEN [22], and APTEEN [23].

Geographic Routing

Most of the routing protocols for sensor networks require location information for sensor nodes. In most cases location information is needed in order to calculate the distance between two particular nodes so that energy consumption can be estimated.

Since, there is no addressing scheme for sensor networks like IP-addresses and they are spatially deployed on a region, location information can be utilized in routing data in an energy efficient way, which we call geographic routing protocols. For instance, if the region to be sensed is known, using the location of sensors, the query can be diffused only to that particular region which will eliminate the number of transmissions significantly. Examples of geographic routing protocols include MECN [28], SMECN [17], GAF [36], and GEAR [39].

RPL (Routing for low-Power and Lossy networks) [35]

Basic mechanism. RPL is standardized by Internet Engineering Task Force (IETF) ROLL group, with the goal of being a sensor network routing protocol for low power and lossy network. RPL is a distance vector routing protocol based on IPv6. RPL constructs Destination Oriented Directed Acyclic Graph (DODAG) considering a scenario that the traffic pattern is convergecast, where edges in DODAG form paths toward a sink node. The sink node initiates the construction of the DODAG by transmitting a DODAG Information object (DIO). This control message is transferred down the network to build DODAG. If a node receives DIO message, a node chooses its parent node and computes its own rank using some objective function, after which it sends the DIO message including rank information to its neighbor nodes. Using this rank information, each node determines the routes to the sink node by forwarding the packet to the node whose rank is lowest among the neighbor nodes.

Low power. RPL adapts the sending rate of DIO messages dynamically. In a network with stable links the control messages are rare, whereas an environment in which the topology changes frequently will cause RPL to send control information more often.

Lossy links. In WSNs, it is possible that good links often rapidly become bad links due to deployment of nodes in harsh environments, nodes often fail to communicate with their neighbors due to consistent interference. In this case, the nodes should be able to change the routes by selecting other node as its parent node, i.e., exploiting spatial diversity for reliability. The nodes maintain multiple potential parents, and are able to switch quickly to available nodes whenever a problematic situation occurs.

3.2 Wake-up Scheduling

3.2.1 Motivation and Requirements

Motivation

Energy-efficiency is one of the primary concerns in WSNs. However, time-urgency often becomes important in some applications, e.g., fire alarm application based on automatic temperature sensing. In particular, in duty-cycled WSNs, which are

a typical way of saving energy, unless data is delivered over many hops under one wake-up period, it would take much time for data to arrive at a sink node. In other words, over each hop a sender has to hold the transmission and wait until the receiver wakes up based on its schedule. Thus, combating against low latency in duty-cycled WSNs is important and highly challenging, where smart control mechanism should be involved for smaller latency, co-working with MAC and routing.

One way of reducing long latency in duty cycled WSNs is by appropriately scheduling wake-up times of nodes, rather than waking up at the same time, considering the structure of routing. This idea is often called as *pipelining* of wake-up times, where the basic idea is to schedule the different wake-up times for different nodes, so as to they are slightly shifted in the sequence of their "depths" with respect to the target sink node in the routing paths.

Requirements

We now summarize the requirements of a pipelining mechanism in what follows:

- *Time synchronization:* In wake-up scheduling, both sender and receiver nodes need to wake up at the scheduled time simultaneously, thus requiring time synchronization. However, local synchronization may be sufficient, if a node knows the wake up times of its neighbors, enabling to know the right time to transmit packets, then the packets from sensor nodes can be delivered to a sink node sequentially. Refer to Section 3.3 for various ideas for time synchronization.
- *Traffic pattern and its routing:* In Section 3.1, we briefly discussed the diverse traffic patterns depending on application-specific tasks and the resulting application-specific routing algorithms. In many cases, in particular, for time/event-driven traffic, a notion of *data gathering tree* is formed to route the packets from sensors to sinks, where sink nodes collect the sensing data from sensor nodes, see e.g., Fig. 9. There, sensing data delivery path forms a uni-directed tree structure from sensor nodes to sink nodes. The idea of scheduling wake-up times through a pipelined structure is best fit to such data gathering tree, and it is not very adequate for query-driven traffic. For example, data-centric routing protocols such as directed diffusion (see Section 3.1.1) may frequently change the data path from sensors to sinks over time due to its reinforcement phase.
- *Mobility restriction:* Once sensor nodes are deployed, a wake-up scheduling, which specifies when each node wakes up, is determined based on the given deployment. Therefore, if nodes are highly mobile, then the schedule should be recalculated by the changed routing paths. This implies that this wake-up scheduling may not be very appropriate for highly mobile sensor nodes, e.g., sensors are attached to wild animals.

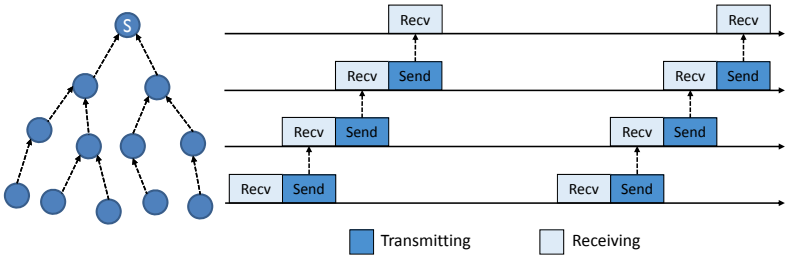

Fig. 9 Pipeline Scheduling in D-MAC

3.2.2 Examples

D-MAC [20]: Pipeline Scheduling

D-MAC [20] was proposed as a MAC to realize the idea of pipelined scheduling of wake-up times at the MAC layer. Once the nodes are deployed, they do not have mobility during a reasonable time and a given routing algorithm forms a data gathering tree, D-MAC schedules the wake-up times of each node based on the depth of the formed data gathering tree. Fig. 9 pictorially shows the pipeline scheduling of D-MAC. Nodes with the same depth will be scheduled at the same time slot. Therefore, those nodes should compete for the channel when they wake up. To reduce collision during this time, each node at the same depth conducts a random back-off within a contention window at the beginning of a scheduled slot. In this way, D-MAC allows sequential packet forwarding from sensor nodes to the sink node, so as to reduce the delay. Similarly to D-MAC, there exist other researches, e.g., fast path algorithm [19], and streamline schedule [5].

Pipeline Scheduling with Lossy Links

The pipeline scheduling of D-MAC is based on a *single* pipeline from the routing algorithm. However, this single pipeline scheme works only when the links in the schedule are reliable, whereas in WSNs links' quality often changes due to the harsh environment in the deployed areas and thus links often become unreliable for some time duration. In such a case, the employed routing protocol detects the status of link unreliabilities and reconfigures the routing paths, and thus the wake-up schedule should be also re-configured accordingly. Robust Multi-pipeline Scheduling (RMS) [12] was proposed to tackle such unreliable links by forming multiple pipeline schedules. In RMS, each node is aware of the wake up schedule of its multiple parents, and for sequential packet forwarding, each node should wake up right before one of its parents nodes' wake up schedule. Then,each node decides on the forwarder among its parents, which can minimize the expected delay.

3.3 Time Synchronization

3.3.1 Challenges and Taxonomy

Challenges

The necessity of time synchronization has been addressed earlier in many places throughout this entire chapter, e.g., in MAC, routing, and wake-up scheduling. In the design of a time synchronization mechanism in WSNs, there are various factors and challenges that should be considered as summarized next. First, clocks embedded in the processor of a sensor node tend to be made of low-quality crystals. This results in frequent clock drifts accompanied with a considerable amount of clock skew. In order to maintain time synchronization among sensor nodes, relative difference in the reference clock drift and offset must be minimized. Second, the resources in WSNs are highly limited in terms of energy, processing power, memory, and communication bandwidths. Thus, too many control overheads for time synchronization may not be allowed. Third, communication environments in WSNs are also very restricted such as frequent transmission failures, channel contentions, and asymmetric message delays. These harsh communication environments prevent the time synchronization in other conventional networks from working as expected, requiring a careful design in WSNs.

Taxonomy

- *Internal vs. External:* Internal synchronization achieves time synchronization in the networks without the master time from an external source, e.g. UTC (Coordinated Universal Time), by keeping internal consistency in local clocks of the network. External synchronization exploits the externally-sourced master time to maintain time synchronization.
- *Clock correction vs. Timetable transformation:* Clock correction is the way of synchronizing local clocks in the network to predetermined standard time, such that the same time is adopted by all clocks in the network at any given point, i.e., global time synchronization. However, in timetable transformation method, time information from other nodes is only translated to that of its own timestamp measurement by consulting its prearranged time transformation table which contains information about offset differences of neighboring nodes.
- *Network vs. Subset:* Time synchronization can be achieved either in the network-wide scope or in the subset scope. Subset-scoped synchronization maintains network-wide time synchronization by appointing intersected nodes to translate time information between the subsets.

3.3.2 Existing Methods

- *Timing-Sync Protocol for Sensor Networks (TPSN) [10]:* TPSN is a time synchronization method adopting a two-way message handshake of the traditional Network Time Protocol (NTP). To operate time synchronization effectively in WSNs, TPSN has two phases: (i) the level discovery phase and (ii) the synchronization phase. In the level discovery phase, the root node broadcasts *level_discovery* packets to neighboring nodes which belong to level 1 nodes relatively to the root. Level 1 nodes broadcast *level_discovery* packets further to complete the tree hierarchy. In the synchronization phase, the root node starts synchronization by broadcasting *synchronization_pulse* based on the hierarchy in the direction of leaf nodes. For two connected nodes at different levels, the nodes with low and high levels are called parent and child nodes, respectively. Each pair of the parent node and the corresponding child exchanges time information in two-way handshake, which makes TPSN as sender-receiver synchronization model.
- *Reference-Broadcast Synchronization (RBS) [9]:* RBS minimizes the critical path of time synchronization by exploiting characteristics of broadcast of wireless nature. The root node broadcasts several packets sequentially to neighboring nodes, for which receiver nodes record the time of arrival of each packet and communicate with other receivers to determine the offsets. In RBS, sending and access delay of the root node are ignored because time of arrival is almost the same among the receiver nodes, which in turn minimizes the critical path. RBS is referred as receiver-receiver synchronization model because each receiver communicates with other receivers rather than the sender node.
- *Adaptive Clock Synchronization (ACS) [26]:* ACS is a hybrid approach between the above two methods to reduce a long critical path of TPSN and the exchange overhead between the receivers in RBS, and to add statistical guarantee on errors. Similarly to RBS, the root node first broadcasts several packets sequentially to neighboring nodes. Each receiver calculates relative clock drift by linear aggression and sends the result back to the root after random delay to prevent transmission conflict. With time information from neighboring nodes, the root node calculates the clock skew and re-broadcasts the result.
- *Time-Diffusion Synchronization Protocol (TDP) [29]:* TDP is a time synchronization method designed to provide a common time throughout the entire network within a certain tolerance, being in contrast to TPSN and RBS where multi-hop synchronization is not supported. TDP divides the role of each node to three different kinds; (1) master nodes, (2) diffused leaders, and (3) regular nodes. After a passive phase where no synchronization mechanism occurs, TDP goes into active phase. At the beginning of each pre-determined period of the active phase, election/reelection procedure (ERP) takes place to determine master nodes. In ERP, false ticker isolation algorithm (FIA) excludes nodes which have high-frequency noise clocks or high access delay fluctuations from being selected as a master node or a diffused leader. Load distribution algorithm (LDA) ensures that a node

with higher residual energy is selected as the master node. After the role decision procedure, master nodes "diffuse" a broadcast message to calculate the standard deviation of round-trip delay, and diffused leaders repeat the message while regular nodes just respond to the message. Every node synchronizes its local clock in accordance with standard deviation information it receives.

- *Rate-Based Diffusion Protocol (RDP) [18] :* RDP adopts a similar approach with TDP in the sense that all nodes in the network are synchronized to one standard time. Each node achieves synchronization by flooding its own local clock and maintains it with the average value, which results the convergence of local clocks to the average value of the clocks in the network-wide scope. RDP is sometimes referred as asynchronous diffusion protocol (ADP) if the synchronization process is operated asynchronously.

4 Summary

Sensor networks are becoming extensively used to effectively and autonomously monitor our environment. In this chapter, we provide a brief overview of networking and communication mechanisms that are needed to conduct an energy-efficient delivery of sensing data together with exemplary protocols in the area of MAC, routing, wake-up scheduling and time synchronization. Clearly, there are more lists that should be discussed than those in this chapter. Our focus is on providing the key challenges, requirements, and different design angles, each of which highly depends on target applications, sensing environments, and the available resources in sensors. Often, the afore-mentioned four components should be jointly designed in a coupled manner, so as for the designed network to be collaboratively optimized for better energy efficiency, higher throughput, and lower latency.

Acknowledgements This work is supported by the Center for Integrated Smart Sensors funded by the Ministry of Science, ICT & Future Planning as the Global Frontier Project.

References

1. Akkaya, K., Younis, M.: A survey on routing protocols for wireless sensor networks. Ad Hoc Networks **3**, 325–349 (2005)
2. Akyildiz, I., Vuran, M.C.: Wireless Sensor Networks. John Wiley & Sons, Inc., New York, NY, USA (2010)
3. Brooke, T., Burrell, J.: From Ethnography to Design in a Vineyard. In: Proceedings of conference on Designing for user experiences (2003)
4. Buettner, M., Yee, G., Anderson, E., Han, R.: X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks. In: Proceedings of ACM SenSys (2006)
5. Cao, Q., Abdelzaher, T., He, T., Stankovic, J.: Towards optimal sleep scheduling in sensor networks for rare-event detection. In: Proceedings of ACM Information processing in sensor networks (IPSN) (2005)

6. Cerpa, A., Elson, J., Estrin, D.: Habitat monitoring: Application driver for wireless communications technology. In: Proceedings of ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean (2001)
7. Dam, T.V., Langendoen, K.: An adaptive energy-efficient MAC protocol for wireless sensor networks. In: Proceedings of ACM SenSys (2003)
8. El-Hoiydi, A., Decotignie, J.D.: WiseMAC: An ultra low power MAC protocol for multi-hop wireless sensor networks. In: Algorithmic Aspects of Wireless Sensor Networks, pp. 18–31. Springer (2004)
9. Elson, J., Girod, L., Estrin, D.: Fine-grained network time synchronization using reference broadcasts. ACM SIGOPS Operating Systems Review **36**(SI), 147–163 (2002)
10. Ganeriwal, S., Kumar, R., Srivastava, M.B.: Timing-sync protocol for sensor networks. In: Proceedings of ACM SenSys (2003)
11. Goense, D., Thelen, J., Langendoen, K.: Wireless sensor networks for precise phytophthora decision support. In: Proceedings of 5th European Conference on Precision Agriculture (5ECPA) (2005)
12. Guo, S., He, T.: Robust multi-pipeline scheduling in low-duty-cycle wireless sensor networks. In: Proceedings of IEEE Infocom (2012)
13. Heinzelman, W.R., Chandrakasan, A., Balakrishnan, H.: Energy-efficient communication protocol for wireless microsensor networks. In: Proceedings of the 33rd Hawaii International Conference on System Sciences (2000)
14. Heinzelman, W.R., Kulik, J., Balakrishnan, H.: Adaptive protocols for information dissemination in wireless sensor networks. In: Proceedings of ACM Mobicom (1999)
15. Intanagonwiwat, C., Govindan, R., Estrin, D., Heidemann, J., Silva, F.: Directed diffusion for wireless sensor networking. IEEE/ACM Transactions on Networking **11**(1), 2–16 (2003)
16. Kim, S., Pakzad, S., Culler, D., Demmel, J., Fenves, G., Glaser, S., Turon, M.: Health monitoring of civil infrastructures using wireless sensor networks. In: Proceedings of ACM Information processing in sensor networks (IPSN) (2007)
17. Li, L., Halpern, J.Y.: Minimum energy mobile wireless networks revisited. In: Proceedings of IEEE ICC (2001)
18. Li, Q., Rus, D.: Global clock synchronization in sensor networks. IEEE Transactions on Computers **55**(2), 214–226 (2006)
19. Li, Y., Ye, W., Heidemann, J.: Energy and latency control in low duty cycle MAC protocols. In: Proceedings of IEEE Wireless Communications and Networking Conference (WCNC) (2005)
20. Lu, G., Krishnamachari, B., Raghavendra, C.: An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks. In: Proceedings of IEEE Parallel and Distributed Processing Symposium (2004)
21. Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., Anderson, J.: Wireless sensor networks for habitat monitoring. In: Proceedings of ACM international workshop on Wireless sensor networks and applications (2002)
22. Manjeshwar, A., Agrawal, D.P.: Teen: a routing protocol for enhanced efficiency in wireless sensor networks. In: Proceedings of IEEE Parallel and Distributed Processing Symposium (2001)
23. Manjeshwar, A., Agrawal, D.P.: Apteen: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks. In: Proceedings of IEEE Parallel and Distributed Processing Symposium (2002)
24. Memsic: MICA2 motes. http://www.memsic.com/
25. Memsic: MICAz motes. http://www.memsic.com/wireless-sensor-networks/MPR2400CB
26. Palchaudhuri, S., Saha, A.K., Johnson, D.B.: Adaptive clock synchronization in sensor networks. In: Proceedings of ACM Information processing in sensor networks (IPSN) (2004)
27. Polastre, J., Hill, J., Culler, D.: Versatile low power media access for wireless sensor networks. In: Proceedings of ACM SenSys (2004)
28. Rodoplu, V., Meng, T.H.: Minimum energy mobile wireless networks. IEEE Journal on Selected Areas in Communications **17**(8), 1333–1344 (1999)
29. Su, W., Akyildiz, I.F.: Time-diffusion synchronization protocol for wireless sensor networks. IEEE/ACM Transactions on Networking **13**(2), 384–397 (2005)

30. Sun, Y., Johnson, D.B.: RI-MAC : A Receiver-Initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks. In: Proceedings of ACM SenSys (2008)
31. Sundararaman, B., Buy, U., Kshemkalyani, A.D.: Clock synchronization for wireless sensor networks: A survey. Ad Hoc Networks (Elsevier) **3**, 281–323 (2005)
32. Tan, H.O., Körpeoğlu, I.: Power efficient data gathering and aggregation in wireless sensor networks. ACM SIGMOD Record **32**(4), 66–71 (2003)
33. Tang, L., Sun, Y., Gurewitz, O., Johnson, D.B.: PW-MAC: An energy-efficient predictive-wakeup MAC protocol for wireless sensor networks. In: Proceedings IEEE Infocom (2011)
34. Tolle, G., Polastre, J., Szewczyk, R.: A macroscope in the redwoods. In: Proceedings of ACM SenSys (2005)
35. Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J., Alexander, R.: RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550 (Proposed Standard) (2012)
36. Xu, Y., Heidemann, J., Estrin, D.: Geography-informed energy conservation for ad hoc routing. In: Proceedings of ACM Mobicom (2001)
37. Ye, W., Heidemann, J., Estrin, D.: An energy-efficient MAC protocol for wireless sensor networks. In: Proceedings of IEEE Infocom (2002)
38. Ye, W., Silva, F., Heidemann, J.: Ultra-low duty cycle MAC with scheduled channel polling. In: Proceedings of ACM SenSys (2006)
39. Yu, Y., Govindan, R., Estrin, D.: Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks. Tech. rep., UCLA Computer Science (2001)
40. Zhang, P., Sadler, C.M., Lyon, S.A., Martonosi, M.: Hardware design experiences in zebranet. In: Proceedings of ACM SenSys (2004)