Data Aggregation Scheduling Algorithms in Wireless Sensor Networks: Solutions and Challenges

Miloud Bagaa, Yacine Challal, Adlen Ksentini, Abdelouahid Derhab, and Nadjib Badache

***Abstract*—Energy limitation is the main concern of any wireless sensor network application. The communication between nodes is the greedy factor for the energy consumption. One important mechanism to reduce energy consumption is the in-network data aggregation. In-network data aggregation removes redundancy as well as unnecessary data forwarding, and hence cuts on the energy used in communications. Recently a new kind of applications are proposed which consider, in addition to energy efficiency, data latency and accuracy as important factors. Reducing data latency helps increasing the network throughput and early events detection. Before performing the aggregation process, each node should wait for a predefined time called *WT* (waiting time) to receive data from other nodes. Data latency (resp., accuracy) is decreased (resp., increased), if network nodes are well scheduled through optimal distribution of *WT* over the nodes. Many solutions have been proposed to schedule network nodes in order to make the data aggregation process more efficient. In this paper, we propose a taxonomy and classification of existing data aggregation scheduling solutions. We survey main solutions in the literature and illustrate their operations through examples. Furthermore, we discuss each solution and analyze it against performance criteria such as data latency and accuracy, energy consumption and collision avoidance. Finally, we shed some light on future research directions and open issues after deep analysis of existing solutions.**

***Index Terms*—Wireless sensor network, data aggregation, multi-channels, media access scheduling.**

I. Introduction

WIRELESS sensor networks (*WSN*) is becoming a very important research area, especially with recent advances in embedded systems and wireless communications. A *WSN* is composed of one or multiple base-stations and a set of sensor nodes deployed in an area called *sensing field*. It is designed to gather data from the network to the base-station (or sink) using hop-by-hop communication. Formally, a *WSN* can be viewed as a virtual distributed database, in which users can formulate abstract requests, and the network protocols strive to answer those requests just like a database management system (*DBMS*). In contrast to *DBMS*, in which the data is stored in the disk and the user extracts these data by sending requests, the request in *WSN* is sent before the data exist and depending on the request the sensor nodes extract the corresponding data from the environment. Sensor nodes are severely resource-constrained, with limited processing power, storage, bandwidth, and battery energy. They can be used in many applications, such as monitoring unreachable and hostile areas. Replacing these batteries requires network redeployment, which can be a very expensive process. As radio communication is the major source of energy consumption [1], one solution to optimize this consumption is to minimize the communication overhead through data aggregation. Instead of sending all individual nodes' readings to the base-station, the raw data are aggregated by intermediate nodes into a reduced number of data packets containing useful information. Data aggregation aims to eliminate redundant packet transmissions by filtering repeated and unnecessary data readings and thus cut on the energy used in communication.

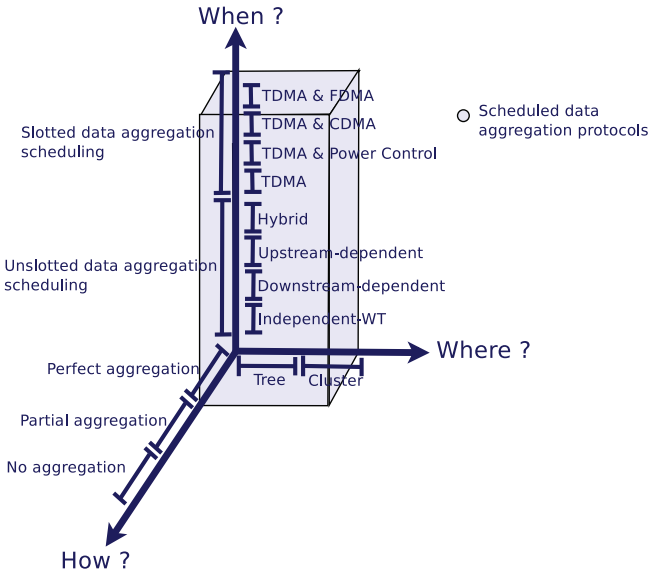


Fig. 1. Data aggregation features

WSNs have numerous applications such as, early detection of forest fire [2], security surveillance [3], and real-time target tracking [4]. These applications can be classified into two categories (i) event-driven and (ii) continuous data collection.

Manuscript received January 22, 2013; revised October 8, 2013 and January 5, 2014.

M. Bagaa and N. Badache are with Research Center on Scientific and Technical Information (CERIST), Algiers, Algeria (e-mail: {bagaa, badache}@mail.cerist.dz).

Y. Challal is with Ecole nationale Supérieure d'Informatique Laboratoire des Méthodes de Conception des Systèmes Alger, Algérie (e-mail: y_challal@esi.dz).

A. Ksentini is with IRISA, University of Rennes 1, Rennes, France (e-mail: adlen.ksentini@irisa.fr).

A. Derhab is with the Center of Excellence in Information Assurance (CoEIA) King Saud University Riyadh, Saudi Arabia (e-mail: abderhab@ksu.edu.sa).

Digital Object Identifier 10.1109/SURV.2014.031914.00029

TABLE I
DIFFERENT DATA AGGREGATION COMPONENTS COVERED BY THE PREVIOUS SURVEYS

Earlier surveys	Components covered
Fasolo et al. [5]	Routing scheme and aggregation functions
Akkaya et al. [6]	Routing scheme and unslotted data aggregation scheduling
Rajagopalan et al [7]	Routing scheme
Cheng et al. [8]	Routing scheme and aggregation functions

In event-driven applications, it is very critical to deliver alarms about serious events in a timely manner so that an appropriate action can be taken in response. Meanwhile, in continuous data collection, it is important to provide a guarantee on the delivery time as well as increase of data collection rate and/or number of nodes in the network. Therefore, the network design should strive to minimize the delay when disseminating the data to the base-station. In-network data aggregation helps in reducing medium access contention as well as the number of transmitted packets, and thus can help in minimizing packet transmission delays.

In contrast to the traditional routing, the intermediate nodes in an aggregation protocol: (i) must be able to access the content of packets to aggregate data; (ii) must wait for a predefined time, called waiting time (*WT*), before aggregating and forwarding data to its upstream node. The waiting time is application dependent and used to wait for all meaningful data before carrying out the aggregation operation. Furthermore, in traditional routing, the delay to receive the first (resp., last) packet by the base-station is proportional to the number of hops between the base-station and the nearest (resp., furthest) node. On the other hand, in data aggregation protocols the delay to receive aggregated values by the base-station is proportional to the duration of waiting time at each node.

Following a deep analysis of the literature, we have concluded that a data aggregation protocol should have three components as depicted in Fig. 1. These components are: (i) an appropriate aggregation function, which specifies how data are aggregated; (ii) routing scheme, which defines how the aggregated data are routed towards the base-station through a structure (tree or cluster); (iii) the aggregation schedule, which defines how long a node has to wait before sending its data to its upstream node. Most of existing solutions aim to optimize one of these components while assuming that the two others are predefined. In this paper, we survey the data aggregation scheduling solutions that aim *to schedule the aggregation process for a given structure and a given aggregation function.* In this taxonomy, our analysis focuses on how far the nodes are scheduled to achieve data aggregation objectives, such that the data latency is minimized and the aggregation freshness is ensured (i.e, data of two generations must not be aggregated together).

There are two kinds of data aggregation scheduling protocols, depending on the *waiting time* (WT) nature. In the first category, the *WT* is continuous with a predefined length. During the *WT*, the node receives the data from its downstream nodes. When the *WT* expires, the received values are aggregated and forwarded. Generally, this category works at routing layer, and it is known as *unslotted data aggregation scheduling* protocols. Usually, the protocols in this category use a contention-based MAC protocol, such as CSMA-based solutions [9], [10]. In the second category, *WT* is subdivided into a set of slots with gaps between them to avoid signal interference. Each slot allows a node to receive or transmit one packet. This category is called *slotted data aggregation scheduling*, and works at both routing and *MAC* layers. The MAC protocol proposed in this kind of solutions is contention-free (i.e., TDMA-based) [9], [10]. Therefore, the protocols in this category assume that the network nodes are well synchronized.

Some surveys on data aggregation in *WSN* have been published in the literature [5], [6], [7], [8]. Fasolo et al. in [5] focused on the study of two components of a data aggregation protocol, which are routing scheme and aggregation functions. They have presented the routing scheme as the problem of forwarding packets in order to facilitate in-network data aggregation. Therefore, in this survey the authors have: (i) presented and discussed solutions that reduce energy when routing the aggregated data; (ii) classified and discussed some aggregation functions existing in the literature. Akkaya et al. [6] focused their study on the impact of data aggregation on data accuracy and latency. For data latency, the authors have showed how this one can be affected by the routing scheme and network nodes synchronization. For data accuracy, they have showed how some unslotted data aggregation scheduling protocols affect the data accuracy. Rajagopalan et al. in [7] evaluated the routing scheme of a data aggregation protocol. The routing protocols were evaluated in terms of energy consumption and network lifetime. Cheng et al. [8] published a survey on data aggregation technologies of wireless multimedia sensor networks (*WMSN*). The authors summarized the data gathering process in *WMSN* in three steps, which are: (i) data acquirement from the environment; (ii) data processing; (iii) data routing. In the first step, some techniques of data acquirement from the literature were described, such as network coverage. In the second step, some solutions of data processing that remove data redundancy, such as data suppression, were presented. In the last step, some solutions to collect the gathered data to the base-station were presented.

Usually, critical applications cannot react to changes of the environment quickly enough to be effective in case of late reporting of aggregated data. In these cases, in-network data aggregation should not only be performed in an energy-efficient way but should also consider other parameters such as data latency and accuracy. As seen in Table I, previous surveys focused on two components of a data aggregation solution which are: the aggregation function and the routing scheme. They showed the impact of these components on the energy consumption and network lifetime. In contrast to the previous surveys, the present paper gives a taxonomy of data aggregation scheduling solutions which aim to manage the waiting time of network nodes. Data aggregation scheduling solutions consider, besides energy consumption, aggregation freshness, data accuracy and latency as critical metrics. In this paper, data aggregation scheduling protocols are classified according to *WT* nature into two categories: (i) unslotted data aggregation scheduling protocols; (ii) slotted data aggregation scheduling protocols. Further, each category is sub-classified

based on its objectives. Some data aggregation scheduling protocols aim to increase data accuracy, whereas the others aim to reduce data latency. To the best of our knowledge, this survey is the first work that proposes a clear taxonomy of data aggregation scheduling protocols in *WSN*.

The rest of the paper is organized as follows: section II discusses the data aggregation protocol features and design goals. Section III discusses and classifies the data aggregation scheduling protocols. The first category, named the unslotted-based data aggregation scheduling, is presented in section IV. The second one, called the slotted-based data aggregation scheduling, is presented in section V. In section VI, we discuss open issues and future directions for data aggregation scheduling protocols. Finally, we conclude this paper in section VII.

II. Data aggregation components and design goals

A. Data aggregation goals

In this section we present design goals of data aggregation protocols and the impact of the waiting time on each goal.

Generally speaking, an aggregation protocol should achieve five main objectives, which are:

*1) **Energy efficiency**[6]:* As long as the *WSN* suffers from the energy resource limitation, the main objective of an aggregation protocol is to save the energy consumption by minimizing the amount of transmitted data. The data aggregation scheduling synchronizes the communications between nodes, and allows them to switch their transceivers off most of the time which leads to save energy.

*2) **Reducing data propagation latency**[11]:* The data propagation latency is considered as an important factor in many applications, such as security surveillance [3] and real-time target tracking [4]. In order to ensure the network scalability for periodic applications, a data aggregation protocol should be able to gather the data from the network whatever the number of nodes and/or frame length. For event driven applications, when nodes detect an event, some alerts should be early generated, aggregated, and forwarded to the base-station. The data aggregation scheduling has a great impact on data propagation latency. The data propagation latency is minimized, if the *WT* is minimized at each node. Furthermore, if *WT* is optimally distributed, the data latency can be further reduced.

*3) **Data Accuracy**[5]:* The amount of transmitted data can be efficiently minimized when the aggregation protocol is used. However, the aggregate value can lose some information. In order to ensure the data accuracy, there is a need to increase the number of nodes' readings participating at each aggregation process. To do so, the number of nodes participating in the aggregation process should be increased. The optimal distribution of *WT* over nodes, allows each one to receive and aggregate the readings of its predecessors and hence increases the data accuracy.

*4) **Aggregation freshness**[11]:* The aggregation freshness is ensured, if the data collected during the same time period (i.e., same frame) are aggregated together. In some applications, aggregation freshness may not be important. For instance, when the design goal is to detect an event like a fire in a forest, the application goal is only to detect this event. If an event occurs the alert messages redundancy can be removed using data suppression since aggregation freshness is not important. But, if we need to know in real-time the border area of a fire, aggregation freshness should be ensured. Indeed, in monitoring applications, aggregation freshness is considered as an important goal, due to the requirement for time-stamping data. The data aggregation scheduling is the most important factor that affects the aggregation freshness. If *WT* of a node is longer than the one of its parent, the data of different frames will be aggregated together.

*5) **Avoiding collisions**[12]:* The data aggregation protocol can help avoiding collisions, by minimizing the amount of data to be carried by the network. Also, it can decrease the number of collisions by a better selection of the data path and the adequate data aggregation scheduling. For instance, if the application uses slotted-based data aggregation scheduling protocol, the collision can be removed through *TDMA* (**T**ime **D**ivision **M**ultiple **A**ccess) mechanism.

B. Data aggregation components

To design a data aggregation protocol, we should take into account three main components as depicted in Fig. 1:

- **Aggregation function (How?)**: This component defines how data are aggregated by specifying the aggregation function used by the protocol,
- **Routing scheme (Where?)**: This component defines how the aggregated data are routed towards the base-station by using or creating a network structure,
- **Data aggregation scheduling (When?)**: This component defines how long a node should wait before aggregating and forwarding received data.

The present paper focuses on the aggregation scheduling component. Therefore, a brief discussion on the aggregation function and routing scheme components is given in what follows. For more information about the aggregation function and routing scheme components, readers may refer to the following surveys [5] and [7].

Depending on the type of aggregation function, the data (i.e., inputs) can be aggregated to one or multiples values (i.e., outputs). For example, if the aggregation function is MIN or MAX, the number of outputs is one, whereas if the aggregation function is MEDIA, MODE or RANK, then the number of outputs equals to the number of inputs. Also, data can be aggregated in-network or at the base-station. Data can be aggregated at the base-station when the number of outputs in an aggregation function equals exactly to the number of inputs, whereas when the number of outputs is less than the number of inputs, the data would be aggregated in-network. In in-network data aggregation, each aggregation function minimizes the amount of data by transforming the inputs data into outputs data within a summarization technique. Each sensor node executes a specific aggregation function in order to minimize the amount of forwarded data towards the base-station. Depending on the number of outputs, in-network aggregation functions can be classified into two classes which are:

- Perfect aggregation function: In this kind of aggregation functions, whatever the number of input data, the aggregation function must culminate into one packet. For

example, MIN, MAX, SUM, and COUNT functions are all perfect aggregation functions;

- Partial aggregation function: In this kind of aggregation functions, the number of output data is more than one, but remains less than the number of input data. For example, *AVG()* is a partial function which is computed using two perfect functions: *SUM* and *COUNT*;

The routing scheme component defines how the aggregated data are routed towards the base-station through a specific network structure. Generally, tree or cluster structures are created as routing schemes. The solutions in these approaches try to create a specified structure, which can be exploited later by the data aggregation protocols. The use of such structure leads to reduce the broadcast in the network; each node sends packets to its upstream which will be routed towards the base-station. Therefore, the data latency and energy consumption will be reduced.

When a cluster structure [13] is used as routing scheme, the network is subdivided into a set of areas called clusters. Each cluster is composed of a set of nodes, one of them is the cluster head and the others are the members. All cluster members send their data to the cluster head. The latter aggregates and forwards the aggregate result to its upstream node. According to the upstream node, there are two kinds of cluster structure. If the cluster heads are one-hop away from the base-station, the solution is said to be one-hop clustering. In this case, each node is at most two hops away from the base-station. The second category is multi-hop clustering, where the upstream of a cluster head is one of its cluster members, called gateway. The latter forwards the aggregate result to another cluster head or to the base-station.

The tree structure is well known in the literature, specially in routing protocols. Many kinds of tree structures are proposed in the literature. In this paper we present only Connected Dominating Set tree and spanning tree, in order to facilitate the presentation of data aggregation scheduling protocols that are surveyed later in this paper and rely heavily on these two structures.

a) CDS (Connected Dominating Set)-based structures [14]: Let G be a network graph which is composed of N nodes. A connected dominating set $S \in N$ is a set of nodes communicating with each other without using nodes in $N-S$. Moreover, any node in $N-S$ has at least one neighbor in S. Thus, all the nodes in $N-S$ can forward their data to a node in S, which can aggregate and forward the data to the base-station. This means that the connected dominating set S can behave as a virtual backbone to gather and aggregate data from N to the base-station.

b) Spanning tree protocol (STP) [15]: The solutions in this category aim to create a spanning tree to save energy consumption. The spanning tree is well known in the literature, as well as its variants, such as the shortest path tree and balanced shortest path tree. For further information about the use of *SPT* in the aggregation protocol, the reader may refer to the survey cited in [7].

In what follows, we will provide a taxonomy and thorough presentation of data aggregation scheduling solutions along with analysis.

III. Data aggregation scheduling

The recent advances in technology allowed the development of low-cost and low-power miniature sensors, such as Cyclops camera [16], which can be integrated into existing motes (sensors) like *Micaz* and *Imote2* [17]. Therefore, many applications have been proposed later, which are greedy to the network bandwidth, such as security surveillance [3] and real-time target tracking [4]. Generally, a sensor node is equipped with a weak transceiver, like *CC2420* [18], which does not provide a large bandwidth. For this reason, the network design should strive to minimize the delay in disseminating the data to the base-station. As explained before, the data aggregation reduces the number of packets in the network, and hence it saves energy consumption and reduces the data latency. However, the waiting time *WT* of each node before aggregating data can significantly affect the data latency and the aggregation freshness. Because of this, many data aggregation scheduling solutions are published in the literature to distribute the *WT* over network nodes, such that latency is minimized and freshness is guaranteed. In this section, our taxonomy and the different performance criteria used to evaluate the existing solutions are presented.

A. *Classification and taxonomy*

As mentioned earlier, the data aggregation scheduling has the strongest impact on data aggregation goals. An optimal scheduling of data aggregation should lead to save energy, increase data accuracy, and ensure aggregation freshness. As depicted in Fig. 2, we classify the data aggregation scheduling into two categories: (i) unslotted data aggregation scheduling and (ii) slotted data aggregation scheduling. In unslotted data aggregation scheduling, a node receives and aggregates data for a predefined interval of time *WT*. At the end of the *WT*, the aggregated values are forwarded. On the other hand, in the slotted data aggregation scheduling a set of slots are assigned to each node in the network. The last slot is designated to forward the aggregated result, whereas the others are designated to receive data from downstream nodes. When there are gaps between slots, a node can switch its transceiver off. Usually, the latency in slotted data aggregation scheduling is less than the one in unslotted data aggregation scheduling.

The base-station receives nodes' data from its neighbors within its *WT* which is also called the frame of data collection (as depicted in Fig. 3). The base-station receives the aggregate data within one or many frames depending on the application nature. In the case of event-driven applications, when events are occurred the alert messages are aggregated and forwarded within one frame. In this case, the length of the frame should be decreased as much as possible to reduce latency. In the case of continuous data collection, the data are aggregated and received periodically by the base-station within one frame; the duration of each period equals to the length of the frame. In this case the length of the frame should be reduced as much as possible so that the data latency is reduced, and data accuracy and aggregation freshness are increased.

*1) **Unslotted data aggregation scheduling**:* This type of aggregation scheduling has been proposed to overcome the need for network synchronization service. In this category, the

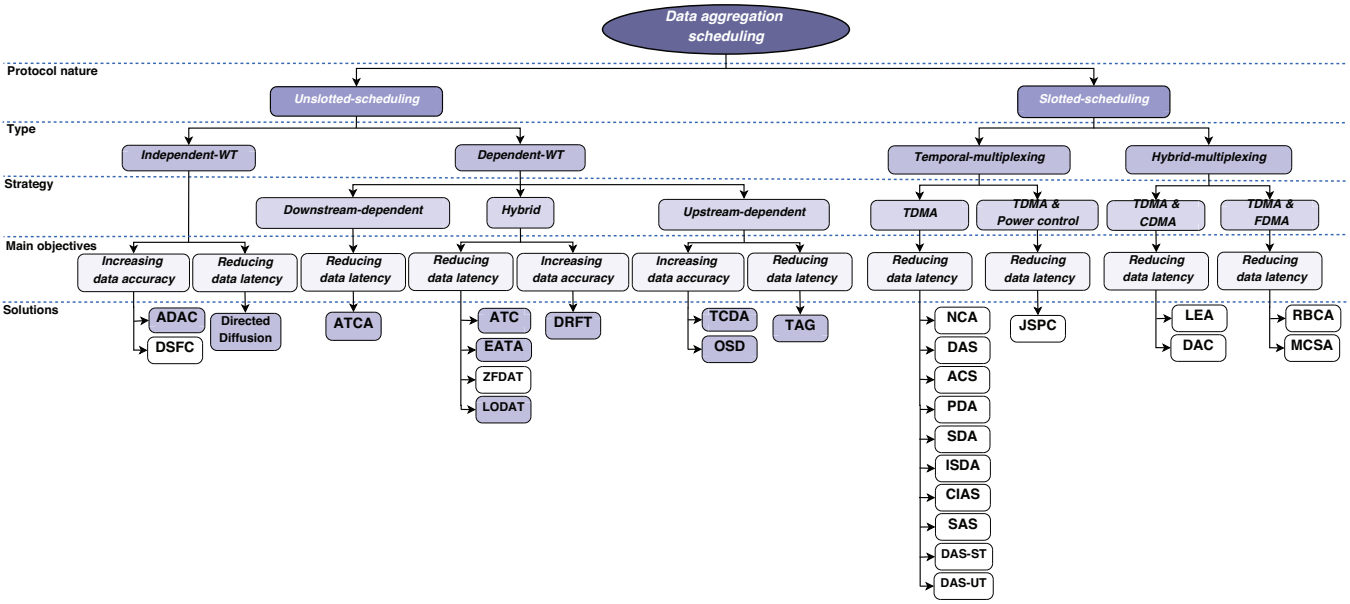


Fig. 2. Data aggregation scheduling taxonomy

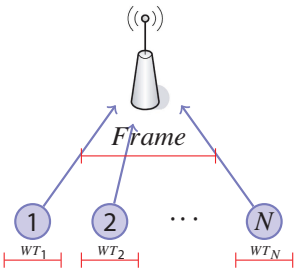


Fig. 3. Frame and waiting time

clock drift between nodes is permitted, and nodes synchronization is only done by exploring data messages. The protocols using this type of scheduling, allow each node to switch its radio on only for a precomputed interval of time. During this interval, a node receives data and forwards the aggregated result to its parent. As depicted in Fig. 4, sensor nodes use wireless medium to communicate. However, this medium limit the communication capability of a sensor node. A node does not have the ability to send and receive packets simultaneously. Therefore, in unslotted data aggregation scheduling solutions, a node receives packets from its downstream nodes during the interval of time and then forwards the aggregate result. Due to the radio propagation nature, when nodes simultaneously transmit packets to their upstream nodes many collisions will occur. To overcome this problem, most of existing solutions in this category use contention-based (i.e., CSMA-based) MAC protocols [9], [10], [19]. Based on the techniques used to distribute the *WT* interval, we classify unslotted data aggregation scheduling into two classes (see Fig. 2):

a) ***Independent-WT****:* In this kind of scheduling all the network nodes have the same waiting time *WT* as shown in Fig. 5(a). During this interval, each node receives and aggregates data from its children. When *WT* is elapsed, each node forwards the aggregated value to its parent. As the nodes are not synchronized and the parent's *WT* is equal to its children ones, there is a high probability that most of children' data are received in different frames. Therefore, aggregation freshness and data accuracy can be significantly biased.

b) ***Dependent-WT****:* This class of protocols enhances the previous one in terms of aggregation freshness and data accuracy. In this class of protocols, the *WT* of different nodes cannot be the same. We classify this class of protocols into three sub-classes according to the way to adjust the *WT*:

- **Downstream-dependent:** In this scheduling model, nodes having the same number of downstream nodes (i.e., children in the aggregation tree) obtain the same *WT* (i.e., interval of time). As depicted in Fig. 5(b), the interval is increased proportionally to the number of children. Thus, the node that has the largest number of children waits more than the others. In this case, the aggregation freshness and data accuracy are better than the ones in independent-WT. However, the aggregation freshness would be affected, if there are some nodes in the network having more children than their children.
- **Upstream-dependent:** This scheduling model [20] aims to bypass the problem of aggregation freshness. The waiting time of each node in upstream-dependent *WT* is related to the number of hops from the base-station. As depicted in Fig. 5(c), the waiting time of each node is increased inversely to its position by report to the base-station. In other words, a node which is closer to the base-station will have a higher waiting time. This strategy allows the parent node to aggregate received data from its children in the same frame, which increases the aggregation freshness. It is worth mentioning that upstream-dependent *WT* and downstream-dependent *WT* are better than independent-WT.
- **Hybrid approach:** The protocols using upstream-dependent *WT*, employ only the hop count information to determine how long a node can wait before aggregating and forwarding received data to its parent. This strategy suffers from two problems:

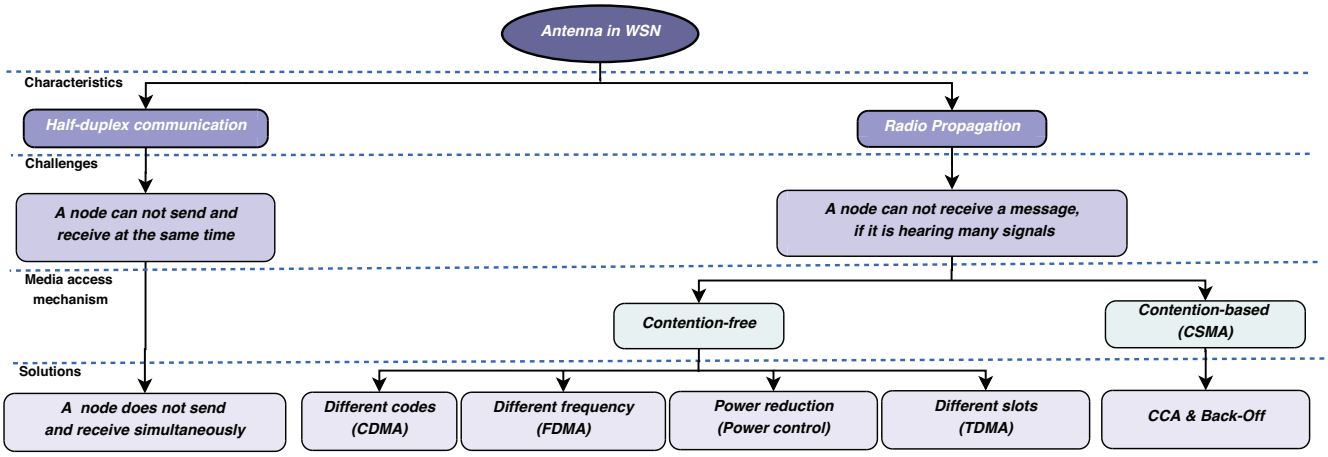


Fig. 4. Challenges in wireless sensor network

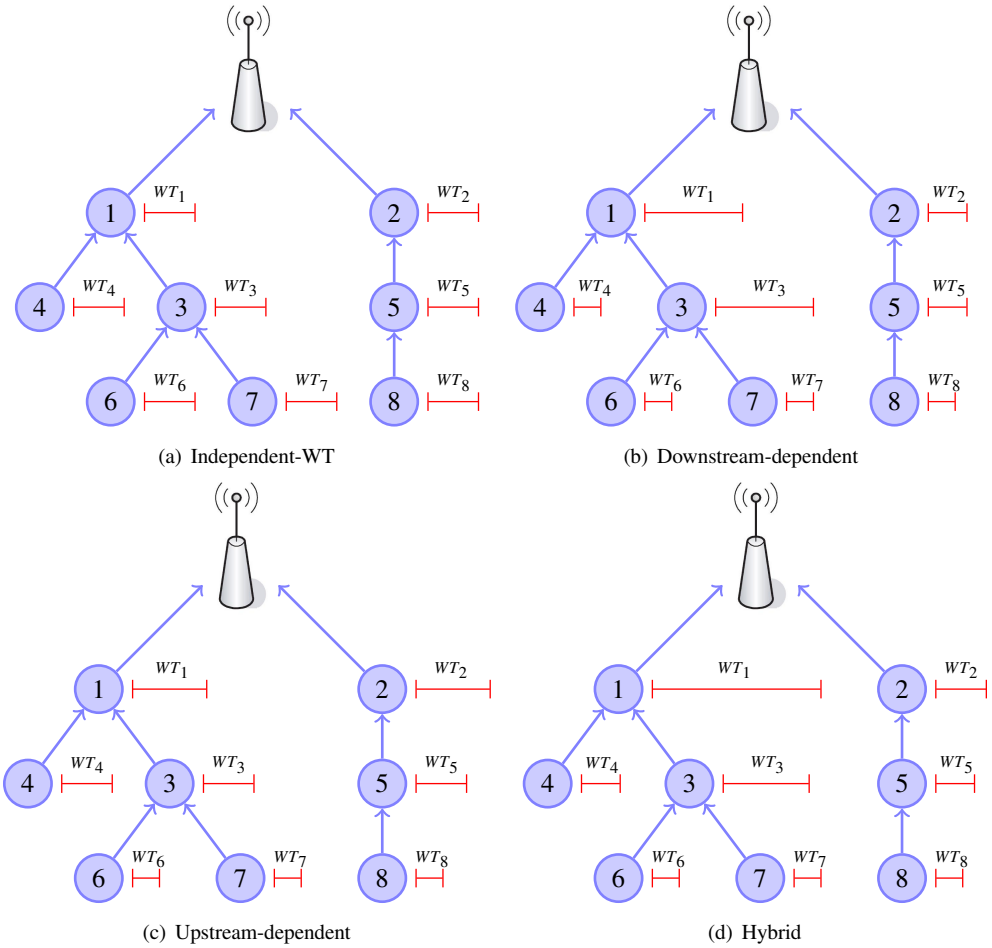
– The nodes in the same level can forward simultaneously the aggregated data to their parents which may lead to many collisions,
– The nodes in the same level may not have the same number of predecessors in the aggregation tree. The aggregator which has a larger number of predecessors can clock-out the aggregated data before receiving all the data from its children. Therefore, the aggregation freshness can be significantly damaged.

To overcome the above issues, a hybrid approach was recently published in [21]. In this approach, WT calculation is based on both hop-count and number of predecessors (i.e., weighted-path) in the tree. Each aggregator node in the data aggregation tree is a root of a subtree, like in Fig. $5(d)$, where node 1 is a root of two sub-branches. The first one contains only node 4, while the second one contains nodes 3, 6 and 7 (i.e., weighted-path$=3$). The waiting time of each node is adjusted according to its weighted-path. The latter is adjusted with regard to: (i) the number of children; (ii) the sub-branch which has the maximum weighted-path. The weighted-path of node 1 is three nodes, whereas the weighted-path of node node 2 is two. As depicted in Fig. $5(d)$, even nodes 1 and 2 have the same hop-count, the waiting time of node 1 is higher than that of node 2. That is, the weighted-path of node 1 is higher than that of node 2. Furthermore, two nodes belonging to different levels should have different *WT*. This difference is called the minimum cascading time between two neighboring levels φ, which can reduce the collision among nodes. As shown in Fig. $5(d)$, despite the fact that nodes 8 and 4 have the same weighted-path, node 4 has a higher waiting time. Thus, if node 4 is a neighbor to node 5, the collision at node 5 will be reduced.

*2) **Slotted data aggregation scheduling**:* Protocols in this category are cross-layer-based, in which the routing and MAC layers are integrated in one framework, such that data latency is reduced and data accuracy is increased. MAC protocol used in these solutions is TDMA (i.e., contention-free) mechanism [9]. Based on the assumption that there is another service which ensures the network synchronization, protocols of this category distribute time slots over network nodes using *TDMA* mechanism. Each time-slot is long enough to receive or transmit one packet. The frame in this category is subdivided into slots, such that the first slot is in the head of the frame, whereas the last one is in the tail. Usually, the aim of slotted data aggregation scheduling is to reduce the data latency. To do that, the length of the frame is reduced as much as possible by reducing the number of slots used in the network. As depicted in Fig. 6, the number of slots used by the network nodes is four slots. In Fig. 6, the blue (resp., red) slot means that a node transmits (resp., receives) a packet to (resp., from) its (i.e., child) parent, whereas a black slot means that a node is in idle mode (i.e., its transceiver is switched off). The *WT* duration in slotted-based approach is less than the one in unslotted-based for the followings reasons:

1) The nodes are synchronized in slotted data aggregation scheduling applications, each node has a predefined time to receive and forward data. Moreover, the use of time-slots allows the network nodes to switch off their transceivers most of the time and avoid simultaneous transmissions by multiple nodes. Therefore, the messages collision are prevented and hence data latency and energy consumption are reduced.
2) Unlike the unslotted data aggregation scheduling approach, in which the time to receive and forward data is continuous, there are gaps that can appear in slotted data aggregation scheduling approach (i.e., idle mode). As depicted in Fig. 6, node 6 in the first slot transmits its data, which are received by node 3, and can switch to idle mode for slots 2, 3 and 4. Node 7 uses the second slot to avoid creating collision at node 3. Therefore, node 7 switches off its transceiver for slots 1, 3 and 4. Then, it sends its data for the second slot.

For the sake of simplicity, in the rest of the paper, for each node we will present only its transmission slot. Fig. 7 shows how the choice of a topology structure can affect the time slot distribution scheme as well as network latency. Fig. $7(b)$ and Fig. $7(c)$ show the impact of cluster and tree structures on time slots distribution, respectively. In this case, four time slots are needed to receive the final aggregate data by the base-station if cluster structure is used. If the tree structure is used,



(a) Independent-WT (b) Downstream-dependent

(c) Upstream-dependent (d) Hybrid

Fig. 5. Different unslotted data aggregation scheduling protocols

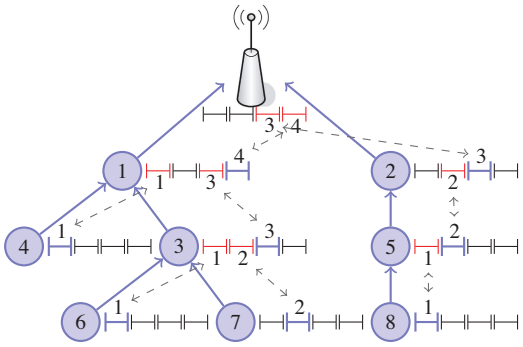


Fig. 6. Slotted data aggregation scheduling: an example

only three slots are required by the base-station to receive the aggregate result. Hence the tree-based topology is the best one as shown in Fig. $7(c)$, since it minimizes the network latency (i.e., 3 time slots per frame). On the other hand, if the time slots are efficiently distributed, a low latency can be achieved. However, the creation of such a structure with the best time slot distribution is proved to be an NP-hard problem in [22].

Generally speaking, there is a trade-off between data latency and aggregation freshness in slotted data aggregation scheduling protocols. On one hand, to reduce data latency the parent should select the smallest time slot which does not create a conflict. On the other hand, if a node selects a time slot which is lower than the one of its child, the aggregation freshness will be affected. Most of existing solutions aim to ensure this trade-off by assigning to the parent node the smallest time slot which is higher than all the ones of its children. This allows the parent to aggregate all children's data together at the same frame. Fig. $8(b)$ shows the data gathering process for two frames when using slots distribution as shown in Fig. $8(a)$, which ensures the aggregation freshness. In this case, three time slots are needed to allow the reception of the final aggregation value by the base-station. Further, the data of each frame are aggregated and received by the base-station in the same frame. For instance, the data a_1, b_1, c_1 and d_1 of the first frame are aggregated and received in this one. Fig. $8(c)$ shows an example of slots distribution which minimizes time latency without taking into account the aggregation freshness. As depicted in Fig. $8(c)$, only two slots are required to forward all the data to the base-station. The data of different frames can be aggregated together. For example, the data d_1 of the first frame and the data a_2, b_2, c_2 of the second frame are aggregated together as shown in Fig. $8(d)$.

Slotted data aggregation scheduling protocols are different in the sense that all of them explore the cross-layer advantage of using *TDMA* mechanism. Therefore, using *TDMA* is mandatory in all slotted data aggregation scheduling protocols. In contrast to unslotted data aggregation scheduling protocols, slotted-based ones take into account (i) the delay of messages

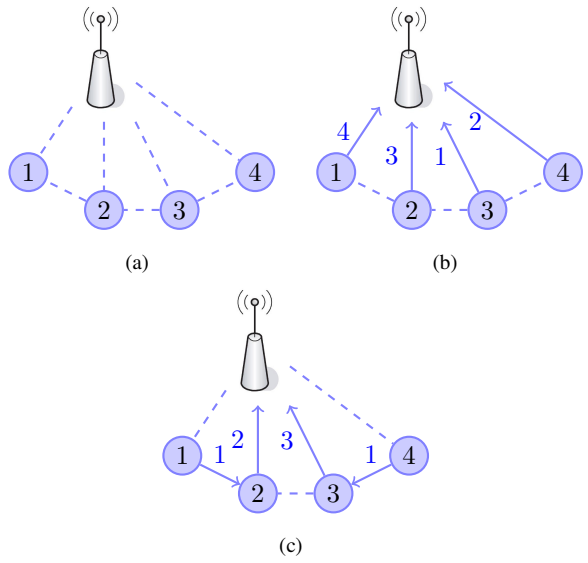


Fig. 7. Impact of data aggregation structure on data delay

transmission and (ii) messages collision. Currently, sensors are equipped with wireless transceivers to communicate. However, due to wireless communication nature, a node does not have the ability: (i) to send and receive messages simultaneously or (ii) to use more than one frequency simultaneously. This limitation leads to extend the challenges at the network nodes in two categories as depicted in Fig. 4:

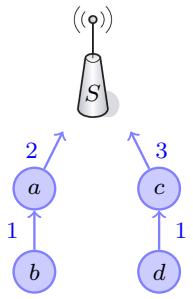
- First challenge: A node cannot send or receive messages at the same time;
- Second challenge: When two nodes simultaneously send messages, then the third one, which is in their interference range, cannot hear them.

To overcome the first challenge, the parent node must send and receive messages at two different slots. Nevertheless, to address the second challenge in contention-free MAC protocol, one of the following techniques are used as depicted in Fig. 4:

1) A code is used for the protocols which use, in addition to *TDMA* schedule, the *CDMA* technique,
2) A frequency is used for the protocols which use, in addition to *TDMA* schedule, the *FDMA* technique,
3) A reduction of node's transmission power for the protocols which use, in addition to *TDMA* schedule, the power control technique.

If the above techniques cannot overcome the second challenge, then each sender must use a different time slots. However, these techniques lead to increase the data latency. Consequently, we divide slotted-based data aggregation scheduling into two classes according to the employed medium access mechanism: (i) *temporal-multiplexing* and (ii) *hybrid-multiplexing*.

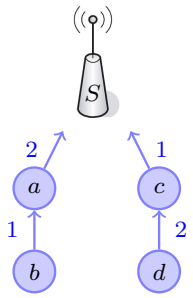
- **Temporal-multiplexing:** The protocols of this category use only *TDMA* technique to access the wireless medium. Further, we sub-classify this class of protocols into two subclasses as depicted in Fig. 5:
  - ***TDMA*:** The protocols [23], [12], [24], [25], [22], [26], [27], [28], [29] of this category aim to reduce data latency by distributing time slots among network nodes. Usually, three techniques are used in these protocols to reduce the data latency: (i) the use of the adequate tree which can reduce the conflict at the parent nodes; (ii) the intertwine of aggregation tree formation and the allocation of time slots (i.e., data aggregation scheduling) to nodes; (iii) the use of heuristics to allocate time slots to nodes, since data aggregation scheduling is *NP-hard* [22].
  - ***TDMA* & Power control:** In addition to the above techniques, existing solutions in this category use another technique to decrease the number of interferences, and hence to decrease the data latency [30]. Before starting the data aggregation scheduling, the power transmission of each node is reduced as much as possible such that (i) a node can communicate with its parent and (ii) interference with other nodes is mitigated.
- **Hybrid-multiplexing:** The protocols [31], [32], [30],



(a)

	Frame1			Frame2		
Receiver	$Slot_1$	$Slot_2$	$Slot_3$	$Slot_1$	$Slot_2$	$Slot_3$
S		a_1, b_1	c_1, d_1		a_2, b_2	c_2, d_2
a	b_1			b_2		
b						
c	d_1			d_2		
d						

(b)



(c)

	Frame1		Frame2	
Receiver	$Slot_1$	$Slot_2$	$Slot_1$	$Slot_2$
S	c_1	a_1, b_1	c_2, d_1	a_2, b_2
a	b_1		b_2	
b				
c		d_1		d_2
d				

(d)

Fig. 8. Trade-off between data freshness and data latency

[33], [34] of this category in addition to *TDMA*, use another technique to access the medium. According to the second technique nature, we sub-classify this class of protocols into two subclasses as depicted in Fig. 5: (i) *TDMA & CDMA* and (ii) *TDMA & FDMA*.

3) Taxonomy: The design goal of the application has an impact on data aggregation objectives. For example, if the design goal of the application is to detect a fire in a forest, then if a fire is occurring, alerts are generated and forwarded from the network nodes to the base-station. In addition, the data suppression procedure is used as an aggregation function to eliminate the data redundancy, since aggregation freshness is not important in this case. Accordingly, data aggregation protocols for this application can ignore aggregation freshness objective. Based on our analysis of existing solutions, a data aggregation scheduling protocol aims to achieve one of the design goals while considering the others as constraints. According to the objectives and constraints, we can classify the data aggregation scheduling solutions into two categories:

1) Protocols that aim to increase the data accuracy: Solutions in this category aim to increase the rate of data which participate in the aggregation process in each frame to exceed a predefined threshold. Usually, the protocols of this category are used in periodic collection applications. Let us denote by N_{opt} the optimal number of packets that should participate in the aggregation process. The length of the frame is varied according to the number of packets participating in the aggregation process of the previous frame and N_{opt}. If the number of packets participating in the aggregation process of the previous frame is less than N_{opt}, the length of the frame will be increased, otherwise it will be decreased. Formally, we can define the objectives and constraints of this category as follows:

$$\begin{cases} \textbf{Objective:}\ increase\ data\ accuracy \\ \textbf{Constraints:} \\ \quad \textbf{(1)}\ reduce\ data\ latency\ and\ increase\ aggregation\ freshness. \\ \quad \textbf{(2)}\ save\ energy\ consumption\ and\ avoid\ messages\ collision. \end{cases}$$

2) Protocols that aim to reduce the data latency: The solutions of this category aim to reduce the data latency while taking into account the data accuracy and the aggregation freshness. Usually, this kind of protocols are used in event-driven applications. A subclass of these protocols, in addition to the data accuracy and aggregation freshness constraints, requires that the data of network nodes should be received by the base-station within a predefined deadline T_{max}. In this case the length of the frame does not exceed T_{max}. We can formally define the objectives and constraint of this category as follows:

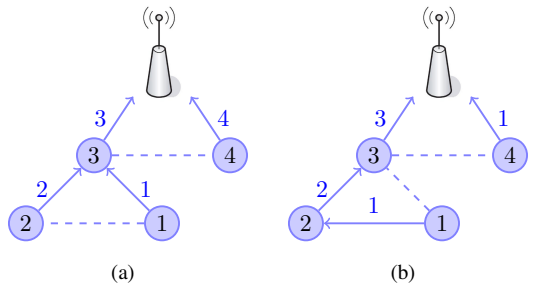


Fig. 9. Impact of parent selection on time latency

$$\begin{cases} \textbf{Objective:}\ reduce\ data\ latency \\ \textbf{Constraints:} \\ \quad \textbf{(1)}\ increase\ aggregation\ freshness\ and\ data\ accuracy. \\ \quad \textbf{(2)}\ save\ energy\ consumption\ and\ avoid\ messages\ collision. \\ \quad \textbf{(3)}\ receive\ the\ aggregate\ values\ before\ T_{max}\ is\ exceeded \end{cases}$$

Surveyed solutions in this paper are depicted in Fig. 2. These solutions are presented with rectangles, where different colors are used according to the employed MAC protocol. Solutions using contention-free (i.e., TDMA-based) MAC protocol are presented with white rectangles, whereas the ones using contention-based (i.e., CSMA-based) MAC protocol are presented with blue rectangles. As depicted in Fig. 2, the different subclasses of unslotted and slotted data aggregation scheduling protocols are also further classified with respect to their objectives into two categories: (i) protocols that aim to reduce data latency and (ii) protocols that aim to increase data accuracy.

B. *Performance metrics and optimization mechanisms*

*1) **Energy saving**:* The energy saving is an important factor for any protocol in *WSN*. Therefore, it is legitimate to consider the energy consumption as the first performance criterion. Usually, the slotted data aggregation scheduling protocols allow nodes to switch to idle mode more often than unslotted data aggregation scheduling protocols. For this reason, the slotted data aggregation scheduling protocols consume less energy than the unslotted data aggregation scheduling ones. Through this paper, the different data aggregation scheduling protocols will be discussed and evaluated in terms of energy consumption.

*2) **Resilience to clock drifts**:* Data aggregation scheduling protocols are executed following two main phases:

- Bootstrap: In this phase the waiting time is distributed over network nodes,
- Data gathering: In this phase the data are aggregated, usually using perfect aggregation function, from the network nodes to the base-station.

To obtain an efficient data aggregation scheduling protocol, data gathering phase should be much longer than the bootstrap one. Since global perfect clock for the network nodes does not exist, the *WT* distribution will be incorrect during the

data gathering phase. Most of unslotted-based data aggregation scheduling protocols use many mechanisms to adjust the waiting time, and thus capture the clock drift during the data gathering phase. Meanwhile, in slotted-based data aggregation scheduling protocols the network nodes are synchronized, and then there is no need for any mechanism to adjust the waiting time during the data gathering phase. Thus, the resilience to clock drifts is an important performance criterion of data aggregation scheduling protocols.

*3) **Collision avoidance**:* Besides collision avoidance considered as one of data aggregation goals, is an important performance criterion for existing protocols. When a solution reduces the number of collisions in the network, it saves the energy consumption and decreases data latency. The unslotted data aggregation scheduling protocols reduce collision by enhancing the *WT* distribution over network nodes. The slotted data aggregation protocols overcome the collision issue by taking into account the different interferences on wireless links. Several interference models have been considered in the literature to simulate the wireless network behavior. It is worth mentioning that the interference models have an impact on all protocols used in *WSN*. This is due to the fact that sensor networks are based on wireless communications, which have their own specifics in terms of interferences and bandwidth. The interference models are widely considered in wireless networks, and hence many models are proposed and used in the literature. In this paper, we consider two models which are widely used in the literature:

- **Interference protocol model [35]:** In this model, a node hearing two messages at the same time, cannot receive them due to interference. Depending on the interference range R_i and communication range R_c, this model is divided into two classes. In the first class, the communication range is equal to the interference range ($\rho = \frac{R_i}{R_c} = 1$). In the second class, the interference range is higher than the communication range ($\rho = \frac{R_i}{R_c} > 1$). The main advantage of the interference protocol model is its ability to use the graph theory to model *WSN*. However, its drawbacks are two folds: (i) First, it cannot reflect the interference in the real world, since real traces indicate that interference is not a binary phenomenon [36]; (ii) Second, this model can be pessimistic, since in the real world a node can hear two messages and receive only one of them, if the interferences are tolerable.
- **Interference physical model [37]:** This model, also known as *SINR* (Signal to Interference and Noise Ratio), tries to model the interferences by introducing the power of transmission and the noise at the receiver node. Thus, a node in this model is able to receive a message if the *SINR* value is more than a predefined threshold.

*4) **Data latency optimization**:* Many mechanisms are used to reduce the data latency when scheduling the network nodes. In this subsection, we present mechanisms that are used by data aggregation scheduling protocols to reduce data latency.

*a) **Network topology**:* the network topology has a great impact on the data aggregation scheduling solutions. A good choice of network topology can reduce data latency, save energy, and minimize message collisions. As mentioned in section II, cluster structure has a negative impact on energy consumption, messages collision, and data latency, while the tree structure has a positive impact on those objectives. Most of the data aggregation scheduling solutions use the second one as routing scheme. As shown in Fig. $7(d)$, the tree structure can reduce significantly the data latency in slotted-based data aggregation scheduling protocols.

*b) **Parent selection criterion**:* a network can be divided into levels such that each level contains a set of nodes with the same hop-count to the base-station. A node i in level L (L is the shortest distance between i and the base-station) has only links with nodes in levels $L-1$ (i.e., next-level), L (i.e., same-level), and $L+1$ (i.e., previous-level). In some solutions, the parent is only chosen from the node's $(L-1)$-level. In other solutions, it is chosen from the node's next-level or the same-level. The latter are likely to offer better waiting time distribution as less relaxed rules are imposed on choosing the parent node. In order to further reduce time latency, some solutions give more flexibility in parent nodes selection. In these solutions, a parent node can be selected from three levels i.e., next-level, same-level, or previous-level. Fig. 9 shows the impact of parent selection criteria on slotted-based data aggregation protocols. In this Figure, we can see the advantage, in terms of minimum time slot (i.e data latency), achieved when a node can choose its parent from level L or $L-1$, in comparison to the one restricted to choose its parent only from $L-1$ level. In Fig. $9(a)$, the node can chose its parent only from $L-1$ level. Node 1 has no choice but to choose node 3 as its parent, it sets its schedule time to 1. As node 3 is connected to nodes 1 and 4, the minimum time slot, i.e., 4, can be used by node 4. In Fig. $9(b)$, slot 1 can be reused by node 4 when node 1 is allowed to choose its parent (i.e., node 2) from level L.

*c) **Simultaneous vs. sequential approach**:* the data aggregation scheduling protocol mainly executes two tasks: (i) routing structure construction; (ii) aggregation scheduling. These tasks are either executed simultaneously or one after another. In the sequential approach, the aggregation scheduling is forced to follow a particular aggregation order imposed by the aggregation tree, which might not lead to minimize the data latency. Meanwhile, in the simultaneous approach, the aggregation scheduling does not depend on a particular tree, and hence a better (child, parent) configuration could be found from a wide range of possible choices. Fig. 10 shows the advantage of using simultaneous approach than sequential one. The network topology used in this example is shown in Fig. $10(a)$. Fig. $10(b)$ and Fig. $10(c)$ show the tree construction and the schedule of constructed tree, respectively. In this case (sequential approach), the least data will be received at the 5^{th} slot. Meanwhile, Fig. $10(d)$ shows that the least data is received at the 4^{th} slot, if the tree construction and nodes scheduling are executed simultaneously.

IV. UNSLOTTED DATA AGGREGATION SCHEDULING PROTOCOLS

In this section, we survey unslotted data aggregation scheduling protocols, which aim to schedule network nodes without the need of network synchronization. They execute

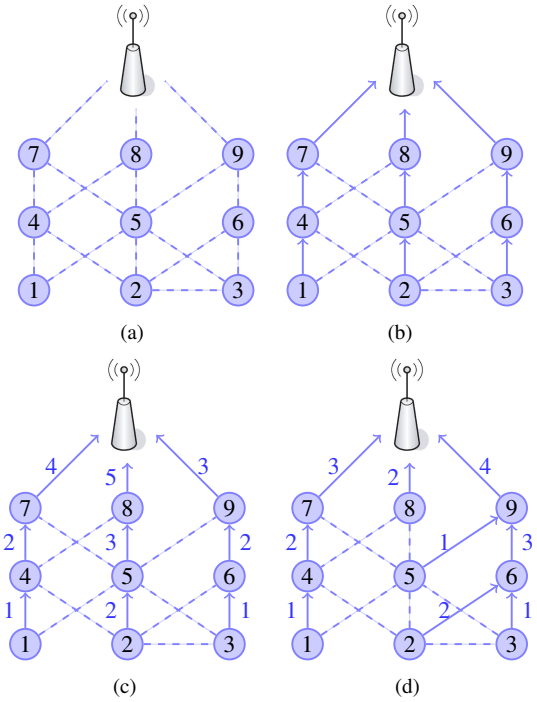


Fig. 10. Impact of execution approach on data delay

mainly two tasks: (i) routing scheme in which a tree or a cluster is constructed; (ii) aggregation scheduling, in which a set of intervals *WT* are distributed over network nodes. All existing solutions aim to minimize data latency and increase aggregation freshness. In data gathering step, these solutions use different techniques to overcome the problem of clock drifts at network nodes. Most of these solutions can be used only in monitoring applications. Fig. 2 summarizes and classifies the reviewed solutions belonging to this category. In table II we introduce the different notations and terminology used throughout this section to describe the different protocols. In the following section, we review solutions belonging to the different classes. We end up this section with discussions highlighting advantages and limitations of the presented solutions.

TABLE II
DIFFERENT DATA AGGREGATION COMPONENTS COVERED BY THE PREVIOUS SURVEYS

Notation	Description
WT_u	waiting time of node u
Frame	specifies the duration of waiting time of the base-station
T_{max}	deadline to receive whole nodes' data by the base-station
T_i	time delay to receive data by the base-station in period i
N_{opt}	Optimal number of packets required by the base-station to achieve data accuracy
N_i	number of packets received by the base-station in period i
τ	the average waiting time of each node in the network
$Nbhop_u$	number of hops of node u to the base-station
$Depth$	depth of the aggregation tree
max_wpath_u	maximum weighted path of node u
min_wpath_u	minimum weighted path of node u
φ	minimum cascading time between two neighboring levels

A. *Independent-WT*

The protocols of this class are classified according to their main objectives into two subclasses which are: (i) the protocols which aim to increase the data accuracy; (ii) the ones which aim to reduce the data latency.

*1) **Increasing the data accuracy**:*

*a) **DSFC**:* using contention-free *TDMA MAC* protocol, a centralized solution, called *DSFC* (Delay Sensitive Feedback Control), has been published in [37]. *DSFC* aims to increase the data accuracy with regard to aggregation freshness and data latency in monitoring applications. The authors assume that the network architecture is one-hop cluster and it is already constructed. At the end of each frame, the base-station (i.e., cluster head) computes the data accuracy and latency. Afterwards, depending on the computed values and tolerable error, the base-station adjusts and distributes the waiting time *WT* to nodes for the next frame. When node's *WT* expires, it aggregates and sends its data to the cluster-head. Due to the use of one-hop cluster, there is only one aggregator in the network, which can ensure aggregation freshness. As mentioned in section III, the waiting time length in one-hop cluster topology is proportional to the number of nodes in each cluster. Thus, the time latency of *DSFC* is proportional to the number of nodes in the cluster which has a higher number of members. Due to the use of *TDMA* mechanism, the messages collision are prevented. *DSFC* can be greedy in what relates to energy consumption, since at each frame, a broadcast message is used to redistribute the *WT* over network nodes. *DSFC* increases the data accuracy by periodically adjusting the waiting time *WT* of each node. However, the data accuracy cannot be optimal since the waiting time is adjusted for the next frame according to the alert messages loss, whereas the number of events in different frames cannot be the same. The use of one-hop cluster architecture and the adaptation of a sequential approach in which the cluster architecture is constructed first then the network nodes are scheduled, significantly increase the data latency in *DSFC*.

*b) **ADAC**: ADAC* (Adaptive data aggregation for clustered wireless sensor networks) [38] is a centralized solution. It assumes that the network topology is already created and it is one-hop cluster. This means that the weight of the network is two hops. The first hop is from any member node to its cluster head, whereas the second one is from the cluster head to the base-station. *ADAC* aims to ensure the data accuracy in monitoring applications. In this solution, the base-station periodically makes a decision according to the received aggregated data. Before receiving the aggregated data by the base-station, they are processed to remove the temporal and spatial redundancy at node level and cluster level, respectively. Whilst spatial redundancy represents the case where many nodes in the same cluster can sense the same event, temporal redundancy is when a node senses the same event many times. In order to save the energy consumption, *ADAC* aims to find the minimum number of packets which should be received by the base-station to achieve the data accuracy. This one is achieved by controlling the removal

of temporal and spatial redundancy. Removal of temporal redundancy is controlled by adjusting the waiting time at each cluster member, whereas removal of spatial redundancy is controlled by adjusting the aggregation ratio at each cluster head. Members of the same cluster have the same waiting time. Since *ADAC* uses a contention-based *MAC* protocol, it does not prevent message collision, and hence it consumes a very important amount of energy. For the same reason as *DSFC*, the aggregation freshness in *ADAC* is ensured due to the use of one-hop cluster structure. However, the use of cluster structure and the adaptation of a sequential approach, have a negative impact on data latency.

*2) **Decreasing the data latency**:*

*a) **Directed Diffusion**:* based on the assumption that all the network nodes are synchronized, *Directed Diffusion* was published in [39]. *Directed Diffusion* is specially dedicated to the monitoring applications. In this solution, one or few base-stations can aggregate data periodically from a set of nodes in the network, called sources. During the aggregation process, the shortest-path tree is created such as the root of the tree is the base-station and the leaves are the source nodes. When a base-station needs to aggregate data from the source nodes, it broadcasts an interest message which contains the average waiting time $\tau = T_{Max}$, where T_{Max} is the maximum delay required by the application (i.e., the frame duration). Each node u receiving the interest message from v: (i) rebroadcasts it and (ii) makes an entry in its interest cache (i.e., routing table). This entry (called gradient) contains the upstream node UP which is v, and the waiting time τ. Here, UP is used to propagate the aggregation result towards the base-station. Note that, the number of gradients at each node can go up by one per neighbor. At the end, many paths can be created to forward data from any source to the base-station. To minimize the energy consumption, only one path for each source is reinforced and used to forward data to the base-station. A node which is in different paths can aggregate the data of different sources. Thus, the data of a source, which is *k-hops* away from the base-station, can be received from the base-station after $k \times \tau$ for the first time. After that, it can be received only after τ. As the source nodes do not have the same distance to the base-station, data of different frames can be aggregated together, and thus the aggregation freshness can be significantly decreased. Clearly, the base-station receives data from its neighbors after τ. However, if a node which is k hops away from the base-station sends its data, it will be received by the base-station after $k \times \tau$. This means that the data latency is affected in *Directed Diffusion*. Due to the use of a contention-based *MAC* protocol and dependent-WT technique, many collisions can occur during the aggregation process. Thus, *Directed Diffusion* can be greedy in energy consumption.

B. Dependent-WT

*1) **Downstream-dependent**:*

*a) **ATCA**: ATCA* (Aggregation Time Control Algorithm) [40] is a distributed solution that aims to minimize the time latency in monitoring applications. *ATCA* aims to: (i) satisfy the application requirement at each time and (ii) overcome the problem of network synchronization. A sequential approach is adopted by *ATCA*, in which it is assumed that the aggregation tree already exists before starting the scheduling process. The base-station informs the network nodes by a simple flooding of a request message, whatever there is a change in the frame duration T_{max}. When a node u receives this message, it updates WT_u according to T_{max} and its $Nbchild$ (i.e., number of children). Periodically, at the end of WT_u, each node u forwards the aggregated data to its parent. In order to reduce the clock drift between nodes during the data gathering phase, WT of each node is periodically adjusted. When a node u forwards a predefined number N of aggregated data, it increases its WT_u. Each node's WT is continuously increased as time goes. Thus, the time delay (T_i) to transfer the data from nodes to the base-station at time period i can exceed T_{max}. In such a case, the base-station broadcasts an overtime message, in order to invite each node to decrease its WT. If a node does not receive the overtime message for N waiting times, it increases its WT. Otherwise, it decreases this one. The WT is increased or decreased by a rate that depends on other factors. The increase rate at each period depends on the number of received of overtime messages, while the decrease rate depends on the difference between T_{i-1} and T_{max}. Due to the use of a contention-based *MAC* protocol and the periodic flooding of T_{max}, a very important amount of energy can be consumed and many collisions can occur in *ATCA*. However, the periodic control of the *WT* minimizes the clock drifts between network nodes. Moreover, the aggregation freshness and data accuracy are slightly increased due to the use of the periodic control.

*2) **Upstream-dependent**:*

*a) **Increasing the data accuracy**:*

i) TCDA: *TCDA* (Timing control during data aggregation operations of distributed sensor networks) [41] is a distributed solution that aims to minimize time latency and ensure data accuracy in monitoring applications. Here, the data accuracy is defined as the optimal number of data N_{opt} that participates in the final aggregation result. The authors assume that the data aggregation tree (*DAT*) is already constructed, and the maximum number *Depth* of hops in *DAT* is computed at the base-station by using a simple flooding message. *TCDA* has two parts, the first one is executed at the base-station, whereas the second one is executed at the network nodes. At each frame i, the base-station adjusts the frame duration T_{max} according to (1) the number of received data N_{i-1} at the previous frame $i-1$, (2) N_{opt} and (3) a predefined threshold δ. If $N_{i-1} < N_{opt} - \delta$ (resp., $N_{i-1} > N_{opt} + \delta$), T_{max} is highly increased (resp., decreased). Otherwise, T_{max} is slowly increased (resp., decreased), if $N_{opt} - \delta \leq N_{i-1} < N_{opt}$ (resp., $N_{opt} < N_{i-1} \leq N_{opt} + \delta$). Afterwards, the base-station broadcasts T_{max} to the network nodes by using multi-hop communications. Each node u receiving T_{max}, computes its waiting time $WT(u)$ as follows:

$$WT(u) = T_{max} - Nbhop(u) \times \varphi$$

where φ is the minimum cascading time between two neighboring levels and $Nbhop(u)$ is the number of hops from the u to the base-station. The clock drifts between network nodes are minimized due to periodic adjusting of T_{max}. The use

of contention-based *MAC* protocol and the periodic broadcast of the frame duration, have a negative impact on: (i) the energy consumption; (ii) the messages collision. Aggregation freshness and accuracy can be degraded due to the weakness inherited from the per-hop adjusted interval use. As the frame duration is adjusted periodically (according to N_{i-1} and N_{opt}), *TCDA* achieves a low efficiency in term of data latency.

ii) OSD: Based on the assumption that the data aggregation framework is a multi-hops cluster, *OSD* (Optimized scheduling for data aggregation in wireless sensor networks) [42] aims to adjust the frame duration of the monitoring application. In *OSD*, the nodes are organized into clusters, and the cluster-heads are organized into a tree. *OSD* focuses only on the communications inter-clusters; how the waiting time *WT* can be distributed over cluster-heads. For the sake of simplicity, we refer to each cluster head as a simple node in the tree. *OSD* is executed on two phases, the setup phase and the data collection one. The authors assumed that the base-station is powerful (its transmission power is high) and can reach the whole network nodes with one broadcast. In the setup phase, the base-station broadcasts a global request to all the network nodes. After the reception of this message by all leaf nodes (i.e., the cluster-heads that have the longest distance to the base-station), each one of them sends a response message to its parent. This message contains a depth field which is initialized by zero. During the response gathering, each parent uses one class of the data aggregation functions. So, when a parent node receives all the response messages from its children, it increments the depth and then it uses one of the following aggregation functions:

1) Perfect aggregation: It sends only the maximum depth to its parent.
2) Partial aggregation: It can group its children by depths.
3) No aggregation: It sends the whole children information to allow the base-station to construct the network topology.

Using the above received information and the optimal number of nodes that must participate in the aggregation process, the base-station can determine the frame duration T_{max}.
During the data aggregation phase, at each frame, the base-station sends T_{max} to all the cluster-heads. Each one computes its waiting time according to its depth in the tree and T_{max}. Throughout the aggregation phase, the authors assume that a perfect aggregation function is used. During the aggregation process, when a parent node receives the whole children data before the maximum aggregation frame expires, it sends its aggregated data to its parent. The aggregated packet contains an extra information to determine the number of nodes participating in the aggregation process, which helps the base-station to recalculate the appropriate T_{max} for the next frame. Note that employing a periodic control of the waiting time, can minimize the clock drifts between network nodes. Further, by using a contention-based *MAC* protocol as well as the periodic control, *OSD* can consume very significant amount of energy and many collisions can occur among the network nodes. As the waiting time of each node is based on its depth in the tree, the data accuracy and aggregation freshness can be significantly decreased in *OSD*. The use of a sequential approach and the cluster structure have a negative impact on data latency in *OSD*. Moreover, the frame duration T_{max} in *OSD* is adjusted according to network state which leads to limit the capability of application layer to specify the frame duration.

*b) **Decreasing the data latency**:*

TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks (*TAG*) is published in [43], where the contributions are:

1) The definition of a simple interface, which allows the end user to formulate its requirement using a SQL-Like declarative language;
2) The distribution and execution of aggregation queries in the sensor network topology.

In what follows, we focus only on the second contribution as the first one is out of the paper's scope. To aggregate data from the network nodes, the base-station computes the average waiting time of each node in the network as follows:

$$\tau = \frac{T_{Max}}{Depth}$$

where T_{Max} is the maximum delay required by the application at each frame and $Depth$ is defined as the maximum number of hops to the base-station in the aggregation tree. Here, it is assumed that the base-station knows the network *Depth*. Afterwards, the base-station broadcasts the request message that contains τ, $Depth$ and the number of hops from the base-station $Nbhop$ (initialized to zero).
When a node u receives a request message for the first time, it:

1) selects the sender as its parent,
2) synchronizes its clock according to the time information in the message,
3) increases by one the received $Nbhop$,
4) sets its waiting time WT_u as follows: $WT_u = \tau \times (Depth - Nbhop)$,
5) updates and broadcasts the request message along with the new hop count.

In *TAG*, during the data gathering phase the network nodes do not adjust their waiting time which increases the clock drifts between nodes. The use of contention-based *MAC* protocol and per-hop adjusted interval technique, have a negative impact on *TAG* in terms of: energy consumption, data latency, messages collision, data accuracy and aggregation freshness.

*3) **Hybrid approaches**:*

*a) **Decreasing the data latency**:*

i) ATC: *ATC* (Adaptive Timing Control) [21] is a distributed solution that considers either monitoring or event-driven applications. The authors assume that each node in the tree knows its hop count $Nbhop_u$ and the number of its children $Nbchild_u$. The aim of *ATC* is to satisfy the delay required by the application T_{max}, and hence ensures that all the aggregated data reach the base-station before T_{max} expires. In order to ensure the aggregation freshness, each node u discards any received packet from its children when its waiting time WT_u is elapsed. For each node u in the network, *ATC* computes the maximum weighted path max_wpath_u and then its waiting time WT_u. To compute the first one, a simple message flooding over data aggregation tree *DAT* is used. When a leaf

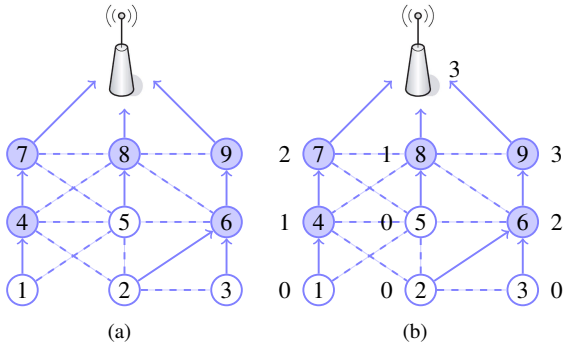


Fig. 11. ATC solution. (a) The network topology such as the direct blue links is data aggregation tree and the white nodes are the leaves

node receives this message, it sends a reply message over *DAT* towards the base-station. The reply message contains max_wpath field, which is initialized to zero at the leaf nodes. After that, each node u in the tree waits to receive the reply messages from all its children. Then, it: (i) selects the maximum value M from its children' max_wpath; (ii) computes its max_wpath_u as $max\_wpath_u = M + Nbchild_u$; (iii) sends reply message containing its max_wpath to its parent. Any node in the network follows the same procedure, until the base-station receives the reply message from all its children. The node sets its $Max\_wpath_{base-station}$ as the maximum value of its children' max_wpath. Finally, each node would have computed its Max_wpath.

To compute the waiting time WT_u, the base-station first computes the average waiting time τ as follows:

$$\tau = \frac{T_{max} - \varphi \times Depth}{Max\_wpath_{base-station}}$$

where φ is the minimum cascading time between two neighboring levels and $Depth$ is the maximum number of hops in data aggregation tree. Then, the base-station broadcasts τ to all the network nodes.

Each node u in the network computes its waiting time based on both $Nbhop_u$ and the number of its descendants. The waiting time WT_u can be computed as follows:

$$WT_u = T_{C_u} + T_{A_u}$$

where T_{C_u} is the cascading time between levels, such that the nodes having the same hop-count belong to the same level. T_{C_u} increases inversely with $Nbhop_u$ to: (i) eliminate messages collision between levels; (ii) allow each node to receive the reading of its farthest predecessors. T_{A_u} is the aggregation time which depends on the number of predecessors, it allows a node to wait for the data from all its predecessors. T_{A_u} increases proportionally with Max_wpath_u. T_{C_u} and T_{A_u} are computed as follows:

$$T_{C_u} = (Depth - Nbhop_u) \times \varphi$$

$$T_{A_u} = \tau \times Max\_wpath_u$$

Fig. 11 shows an example of *ATC*. *ATC* assumes that the data aggregation tree is already created as depicted in Fig. 11(a). In this example the $Depth$ of the tree is equal to three. The aggregation scheduling phase starts in *ATC* by the flooding message from the base-station. When leaf nodes 1, 2, 3 and 5 receive this message, they set their Max_wpath to 0 and then send a reply message whose $Max\_wpath = 0$ to their parents. When the parent node 6 receives reply message from nodes 2 and 3, it selects the maximum values of Max_wpath from received messages, in this case $M = 0$. Then, it sets its $Max\_wpath_6 = M + Nbchild_6 = 0 + 2$, and then it sends a reply message with $Max\_wpath = 2$. Using the same approach, node 9 sets its Max_wpath to 3. As seen above, the base-station sets its Max_wpath as the maximum Max_wpath of its children. As node 9 has the maximum Max_wpath, the base-station sets its Max_wpath to 3. Based on the computed value of $Max\_wpath_{base-station}$, The base-station computes the average waiting time of each node as follows: $\tau = \frac{T_{max} - \varphi \times Depth}{Max\_wpath_{base-station}} = \frac{T_{max} - 3\varphi}{3}$. Afterwards, each node computes its *WT* based on τ value. Let us consider node 9, which computes its WT_9 as follows: $WT_9 = (Depth - Nbhop_9) \times \varphi + \tau \times Max\_wpath_9$, and hence $WT_9 = (3-1) \times \varphi + \tau \times 3$.

To ensure the aggregation freshness in *ATC*, any received data after *WT* is elapsed will be discarded. The discard of data significantly affects the data accuracy in *ATC*, and hence serious alerts cannot be received by the base-station. The network nodes in *ATC* suffer from a significant energy consumption as well as a high number of collisions, which can occur due to the use of contention-based *MAC* protocol. But, the use of weighted-path adjusted interval enhances the data latency in *ATC*. Note that in *ATC*, the clock drifts between nodes can be significantly increased due to the absence of any mechanism to adjust the *WT* during the data gathering phase.

ii) EATA: A centralized solution called *EATA* (EATA: Effectiveness based Aggregation Time Allocation Algorithm for Wireless Sensor Networks) has been published in [44]. It aims to ensure the aggregation freshness and reduce the data latency in monitoring applications. *EATA* assumes that a shortest-path tree is already created. *EATA* tries to capture the trade-off between the aggregation freshness and data latency. Initially, *EATA* takes into account only the data latency. The *WT* of nodes is set close to zero, and hence the time latency is minimized without ensuring the aggregation freshness. Afterwards, *EATA* increases the *WT* of the nodes which has a positive impact on aggregation freshness.

EATA is executed in many iterations. In each one, textitEATA adjusts the *WT* of nodes, which leads to ensure the aggregation freshness with regard to the required delay (i.e., frame duration) by the application. The algorithm ends when a trade-off is achieved between bounded delay and aggregation freshness. At the end of it, the *WT* is increased slightly more for nodes with a higher max_wpath. Finally, the base-station sends the *WT* of the whole nodes using a multi-hop broadcast communication. By employing the weighted-path adjusted interval, *EATA* minimizes the data latency, increases the aggregation freshness and the data accuracy. But, the use of contention-based *MAC* protocol and the multi-hop broadcast communication, have a negative impact on messages collision and hence on energy consumption.

iii) ZFDAT: A distributed solution called *ZFDAT* (Zone-based Fast Data Aggregation Tree) has been published in [45].

ZFDAT aims to satisfy the delays required by the cluster head. *ZFDAT* focuses only on how to distribute *WT* inside each cluster. It can be used in either event-driven or monitoring applications. *ZFDAT* is executed in two steps:

1) The network nodes are organized into a set of clusters,
2) The waiting time is distributed overs each cluster in the network.

In the first step, a set of cluster-heads are randomly elected in a distributed way. After that, each cluster-head broadcasts an announce message using its maximum power transmission to inform the other nodes that it is elected. Each node receiving this message saves the cluster-head identifier and its *RSSI* (Received Signal Strength Indication). Then, each node u chooses its cluster head CH_u as the one with the highest *RSSI* value. In the cluster structure, members that are far from the cluster-head should use more power to send their data. Consequently, they consume more energy than the others. Based on this observation, *ZFDAT* creates a shortest-path tree structure inside each cluster by running the following steps below:

- Each node in the cluster should estimate its position (minimum hop count) to the cluster-head. The authors assume that a node can estimate its minimum hop count to the cluster head based only on: (i) the *RSSI* value of the announce message and (ii) node transmission range. However, as shown in [46] *RSSI* does not allow calculating the distance with high precision, which make the authors' assumption strong,
- The shortest path tree is created inside each cluster, where each node selects its parent as the one whose estimated hop count is less than its own hop count, and then it sends a join-message. Upon receiving this message, each node can deduce the number of its children. At the end, the network nodes are organized into clusters, where in each cluster a *SPT* is constructed to interconnect the leaf nodes with their cluster-head.

In the second step, the waiting time is distributed among the members of each cluster using the same approach as in *ATC* [21]. Furthermore, during the collection of data from the members to its cluster-head, a contention-free *MAC* protocol (*TDMA*) is used to avoid messages collision and data retransmission.

As *ZFDAT* uses the same technique as *ATC* to distribute the waiting time over cluster-members, both of them have the same performances in terms of data accuracy, aggregation freshness and data latency. Nonetheless, the use of weighted-path adjusted interval and contention-free *MAC* protocol, guarantee in *ZFDAT* a low data latency and minimum messages collision. Based on the assumption that the network nodes are synchronized to allow the use of a contention-free *MAC* protocol, the clock drifts between nodes would be prevented.

iv) LODAT: A distributed solution has been published in [47] called *LODAT* (Latency optimized data aggregation timing model for wireless sensor networks) to manage the waiting time for monitoring applications. *LODAT* uses *CTP* (Collection Tree Protocol) [48] to create and maintain the data aggregation tree. The *CTP* protocol uses the wireless link quality to create and update the data aggregation tree. As the tree structure in *CTP* changes frequently, the *WT* of each node is periodically updated in *LODAT*. Initially, the waiting time WT_u of each node u is computed like in *ATC* [21], with little modification on how computing T_{C_u}. In fact, T_{C_u} is computed as follows:

$$T_{C_u} = 2 \times (T_{max} - (\varphi \times Nbhop_u))$$

where φ is the minimum cascading time between two neighboring levels, T_{max} is the delay required by the application and $Nbhop_u$ is the number of hop from u to the base-station. Afterwards, each node u updates its waiting time WT_u whenever there is a change in the tree or reduction in aggregation gain due to time synchronization. The beacon messages are exchanged in regular interval between the nodes. When a node u receives a beacon message, it adjusts its T_{A_u} according to the topology changes and clock drifts. Moreover, for each frame, T_{A_u} is adjusted, increased or decreased by φ, according to the number of received messages from u's children. Due to the fact that *LODAT* employs *CTP* and beacons exchange, many message collisions can occur and hence *LODAT* is greedy with respect to energy consumption. Nevertheless, the periodic update of waiting time and the use of Weighted-path adjusted interval, have a positive impact on data latency, data accuracy, and aggregation freshness.

*b) **Increasing the data accuracy**:*

DRFT: *DRFT* (Delay-ranged feedback timing control) is a distributed solution that aims to increase the data accuracy while taking into account the data latency and aggregation freshness. *DRFT* assumes that the data aggregation tree is already constructed. To distribute the waiting time over network nodes, *DRFT* uses the same technique as *ATC* [21]. The waiting time WT_u of each node u is selected randomly from a set of intervals $S = \{WT_u^{min}, WT_u^{min} + k, WT_u^{min} + 2k, \cdots WT_u^{max}\}$, where k is a predefined constant and WT_u^{min} (resp., WT_u^{max}) is the minimum (resp., maximum) waiting time that can be selected by node u. WT_u^{max} is computed using max_wpath_u, whereas WT_u^{min} is computed using min_wpath_u. In *DRFT*, max_wpath_u and min_wpath_u are computed as in *ATC* [21] with little modifications. To compute max_wpath_u (resp., min_wpath_u), a node receiving reply messages from all its children:

1) Selects the maximum (resp., minimum) value M from its children' max_wpath (resp., min_wpath),
2) Computes its max_wpath_u (resp., min_wpath_u) as the addition between M and its number of children $Nbchild_u$,
3) Sends a reply message containing its max_wpath and its min_wpath to its parent.

Afterwards, each node u in the network computes its WT_u^{min} and WT_u^{max} as follows:

$$WT_u^{min} = \tau \times min\_wpath_u + (Depth - Nbhop_u) \times \varphi$$

$$WT_u^{max} = \tau \times max\_wpath_u + (Depth - Nbhop_u) \times \varphi$$

where φ is the minimum cascading time between two neighboring levels, $Depth$ is the maximum number of hops, and τ is the average *WT* computed as the one in *ATC*. Then, WT_u is selected from $S = \{WT_u^{min}, WT_u^{min} + k, WT_u^{min} +$

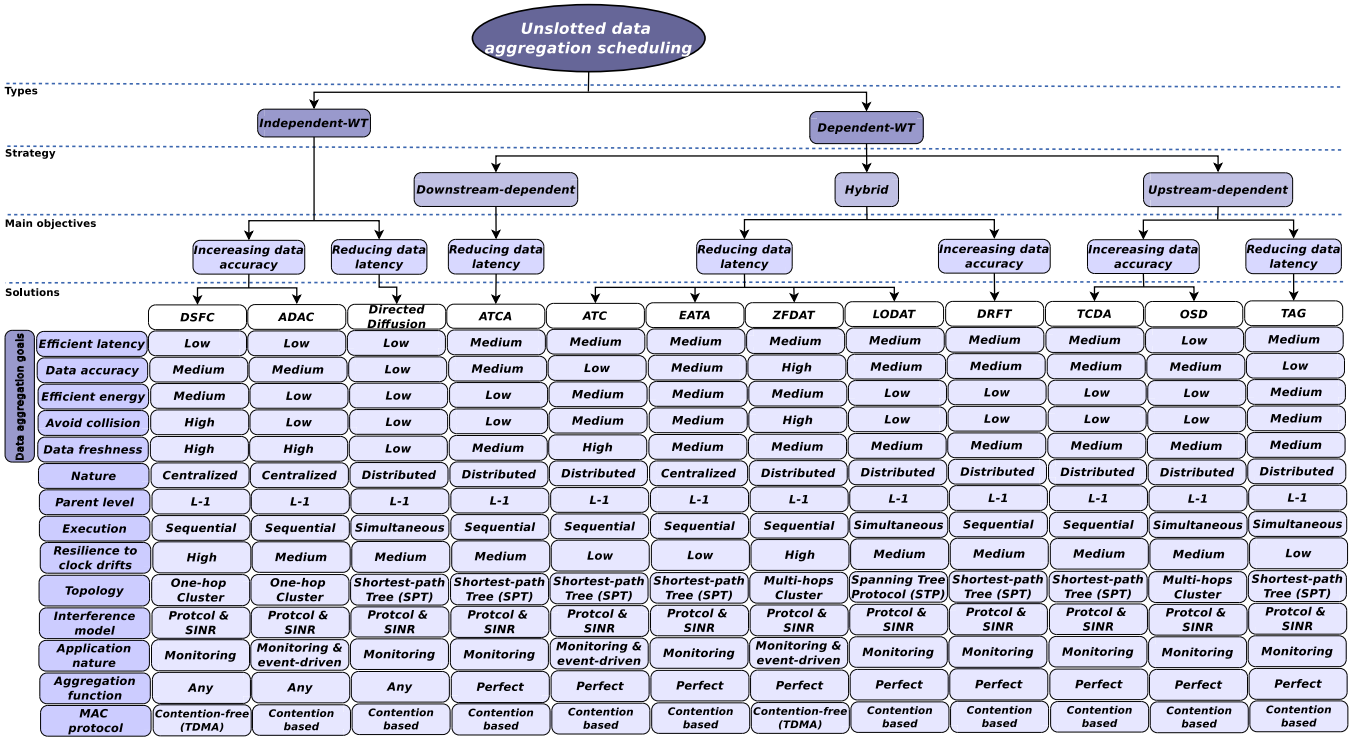


Fig. 12. Unslotted data aggregation scheduling protocols taxonomy

$2k, \cdots WT_u^{max}\}$. Note that, each element in S has the same probability to be selected as waiting time WT_u of a node u.

The number of elements in S is adapted according to the optimal number of sensed data N_{opt} required by the application. According to N_{opt} and the number of received data N_{rec}, the base-station adjusts the number of elements in S for each node. If $N_{opt} > N_{rec}$ (resp., $N_{opt} < N_{rec}$), then WT_u^{max} (resp., WT_u^{min}) is decreased (resp., increased). Note that, the increase of WT_u^{min} leads to an increase of WT_u, which increases the amount of aggregated data. Meanwhile, the decrease of WT_u^{max} leads to a decrease of WT_u, and hence decreases the amount of aggregated data. Because of using a contention-based *MAC* protocol and the periodic update of waiting time, *DRFT* could consume a significant amount of energy and many messages collision can occur. But, the data latency, aggregation freshness and data accuracy are slightly improved due to the use of weighted-path adjusted interval.

C. Discussion of existing solutions

Fig. 12 depicts the taxonomy of unslotted data aggregation scheduling protocols. It gives a comprehensive comparison among the existing solutions in terms of data aggregation goals, algorithm nature, parent level, topology, supported interference model, application nature, aggregation function and *MAC* protocols.

1) **Independent-WT:** As shown in Fig. 12, the protocols of this class are resilient to clock drifts. Independent-WT class contains three protocols, which are *DSFC* [37], *ADAC* [38] and *Directed Diffusion* [39]. Besides using one-hop cluster structure as routing scheme, both *DSFC* and *ADAC* are centralized and sequential. Further, *Directed Diffusion* is distributed, simultaneous and uses *SPT* as routing scheme.
Using a one-hop cluster by *DSFC* and *ADAC* makes those protocols unscalable. *DSFC* outperforms *ADAC* and *Directed Diffusion* in terms of collision avoidance and aggregation freshness, since it uses a contention-free *MAC* protocol. These solutions have a negative impact on data latency as they use an independent-WT and/or one-hop cluster structure. In Dependent-WT protocols, to adjust WT of network nodes, the number of packets transmitted by each node, in each frame, should be known. In order to control the number of transmitted packets, the existing solutions use prefect aggregation function. In contrast to Dependent-WT class, there is no need to adjust WT of network nodes in independent-WT protocols i.e., all the nodes have the same WT. For this reason, in the latter there is no restriction on the aggregation function type.
2) **Dependent-WT:**
   a) **Downstream-dependent:** In this class, there is only one distributed solution *ATCA* [40], which is sequential and resilient to clock drifts. Moreover, it uses *SPT* as routing scheme. The periodic broadcast of overtime message leads to: (i) create many collisions in the network; (ii) consume very important amount of energy; (iii) decrease the time latency efficiency.
   b) **Upstream-dependent:** All the protocols (*TCDA* [41], *OSD* [42] and *TAG* [43]) of this category are distributed. The main objective of *TCDA* and

OSD is the increase of data accuracy, whereas *TAG* aims to reduce the data latency. *TCDA* and *TAG* use *SPT*, whereas *OSD* uses multi-hops cluster structure as routing scheme. The use of multi-hops cluster structure in *OSD* has a negative impact on data latency. In this class of protocols, only *TAG* is not resilient to clock drifts between nodes. In both *TCDA* and *OSD*, the base-station periodically adjusts the nodes' *WT* using a simple flooding messages. It is important to note that, the periodic broadcast used to adjust *WT* has a negative impact on *TCDA* and *OSD* in terms of energy consumption and messages collision.

c) **Hybrid approaches:** There are five protocols in this category, one of them is centralized (*EATA* [44]) and the others (*ATC* [21], *ZFDAT* [45], *LODAT* [47] and *DRFT* [49]) are distributed. The aim of *DRFT* is the increase of data accuracy, while the others aims are to reduce the data latency. Note that, only *LODAT* is simultaneous in this category. As *LODAT* and *DRFT* ensure the resilience to the clock drifts, both of them suffer from high energy consumption and message collisions due to periodic update of *WT*. Although *ZFDAT* avoids the clock drifts, it does not have the same concerns like *LODAT* and *DRFT*, thanks to the use of contention-free *MAC* protocol. All the protocols in this class use SPT tree to reduce the data latency, except *ZFDAT* that uses multi-hops cluster. In order to ensure the aggregation freshness in *ATC*, each node discards the received data from its children when its *WT* is elapsed. However, this strategy affects the data accuracy in *ATC*.

All the unslotted data aggregation scheduling protocols, except for *ADAC*, *ZFDAT* and *ATC*, target monitoring applications as they periodically gather the data from the network nodes. Moreover, in some protocols the network nodes adjust their WT, in each period, based on the information gathered from previous period and/or a received feedbacks from the base station.

V. Slotted data aggregation scheduling protocols

The solutions in this category explore the cross-layer advantage when using *TDMA* mechanism. They execute two tasks: (i) aggregation tree construction; (ii) aggregation scheduling, where a set of slots are assigned to network nodes. It is important to recall that the construction of data aggregation tree with an optimal time slot distribution is proved to be an *NP-hard* problem [22]. During the data gathering step, the data are collected and aggregated from the network nodes by exploring these slots. However, all existing solutions in the literature focus only on the first step; how to create the aggregation tree and distribute time slots among network nodes. Moreover, most of existing solutions assume that, in data collection phase, all nodes should have data to send and a perfect aggregation function is used. Thus, each node in the network must send one and only one packet to its parent. Due to the use of *TDMA* mechanism in data collection phase, all existing solutions tackle the following issues: (i) Messages collision; (ii) Energy consumption; (iii) Resilience to clock drifts between network nodes.

Existing solutions can be used in either event-driven or monitoring applications. Depending on the application nature, the existing solutions could have different objectives to meet. Based on the way to access the wireless medium, we classify the existing solutions in literature into two classes: (i) Temporal multiplexing and (ii) hybrid multiplexing. In the remaining of this section, we present existing solutions belonging to each class and highlight advantages and shortcoming of each one.

A. *Temporal-multiplexing*

*1) **TDMA-based techniques**:* Existing solutions in this category aim to reduce the data latency by exploring the advantage of using cross-layer technique. In order to reduce the data latency, existing solutions use a tree structure instead of cluster one. Other techniques are also explored to reduce further the data latency, such as adopting simultaneous approach and/or increasing more the candidate parents for each node by allowing nodes to select their parents from three levels. Detailed description of these techniques are presented in III-B4. The reduction of the data latency bound is one of the main contribution of TDMA-based data aggregation solutions.

*a) **NCA**: NCA* (Nearly Constant Approximation for data aggregation scheduling) [23] is a centralized solution that aims to minimize data aggregation latency. Based on the hop-count information, the network is divided into levels. Each level contains the set of nodes with the same hop-count to the base-station. By assuming that the base-station is the network topology center (i.e., located in the center of network), the data latency upper-bound achieved by *NCA* is $23R + \Delta - 18$; where R is the network radius and Δ is the maximum degree. A node i in level L (L is the shortest distance between i and the base-station) has only links with nodes in level $L-1$, L, and $L+1$. In order to create the data aggregation tree, a maximal independent set is constructed first. This set is constructed by coloring the maximal independent set of nodes at each level as black, starting from the first level, which only contains the base-station. A maximal independent set at level L must not have a link with maximal independent sets at level $L-1$ or $L+1$. Then, *NCA* picks some nodes from level L and colors them blue in order to interconnect the black nodes in level $L+1$ with the black nodes at level L and $L-1$. The remaining nodes at each level L are colored white, and interconnected to the black nodes at level L or $L-1$. In this way, the desired data aggregation tree is created. After that, *NCA* executes the data aggregation scheduling based on the created tree using the first-fit scheduling procedure. First, all the white nodes are scheduled in such a way that collisions are avoided. Starting from the bottom level, the black and the blue nodes are scheduled level by level using the first-fit scheduling procedure. The latter is executed over iterations, and at each one a set of nodes are scheduled together at the same time slot. A node u is scheduled at time slot τ if it does not have any links with parents of already scheduled nodes at this time slot. However, only the effect of u on the scheduled nodes is

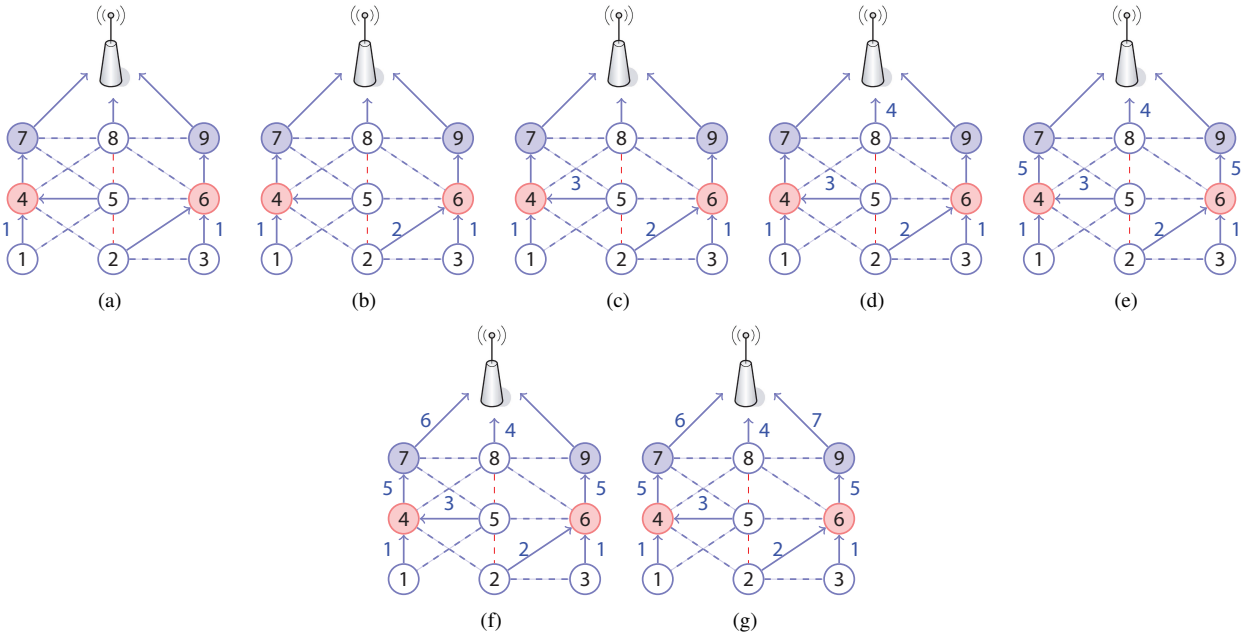


Fig. 13. NCA solution. (a) The network topology such as the direct blue links is the Dominating Set Tree. The white nodes are the leaves, the red nodes are independent dominating set (IDS) and the blue nodes interconnect the IDS together

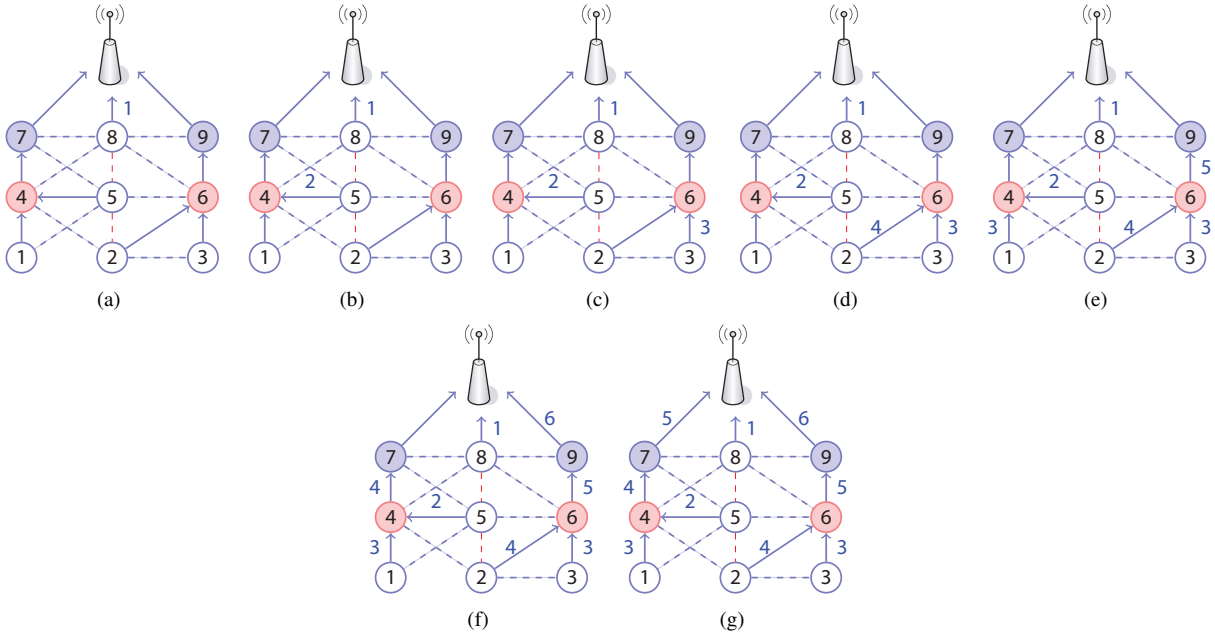


Fig. 14. *DAS* solution. (a) The network topology such as the direct blue links is the Dominating Set Tree. The white nodes are the leaves, the red nodes are independent dominating set (*IDS*) and the blue nodes interconnect the IDS together

tested (there is no test on the effect of the scheduled nodes on u's parent). *NCA* cannot guarantee a collision free schedule in some cases as proved in [12]. Fig. 13 shows an example of *NCA* execution. After the construction of the dominating set tree, the aggregation scheduling phase starts by scheduling the whites nodes, i.e., 1, 2, 3, 5, and 8, as depicted in Fig. 13(a). Nodes 1 and 3 are assigned the same time slot (Fig. 13(a)), the remaining nodes: 2, 5 and 8 are further scheduled (Fig. 13(b) – Fig. 13 (d)). Since all the white nodes are scheduled, *NCA* starts from the bottom level to schedule the black and the blue nodes level by level. The black nodes 4 and 6 are first scheduled (Fig. 13(e)), then the blue nodes 7 and 9 are scheduled (Fig. 13(f) – Fig. 13(g)).

In *NCA*, the leaves are first scheduled, then the rest of nodes are scheduled level by level staring from the node with the largest time slot, which leads to an increase in the data latency. Moreover, as the time slot of the parent node is greater than the ones of its children, the aggregation freshness and data accuracy are ensured in *NCA*.

b) ***DAS****: DAS* (Distributed data Aggregation Scheduling in wireless sensor networks) [12] is a distributed solution for data aggregation scheduling in *WSNs*. This protocol generates

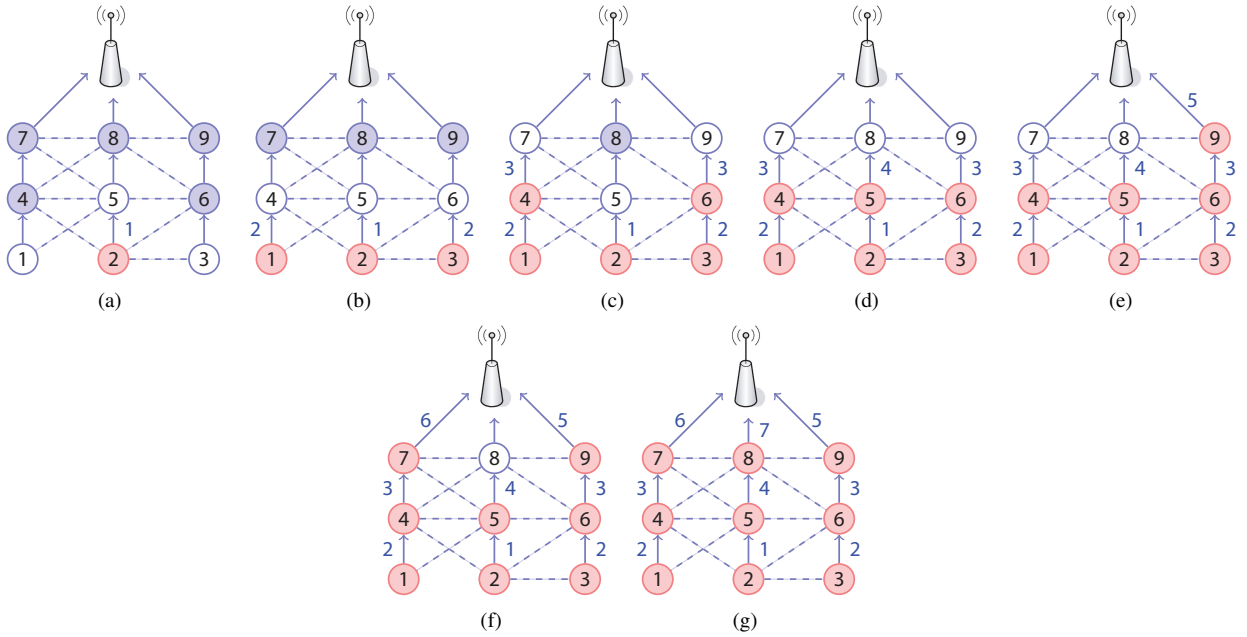


Fig. 15. ACS solution. (a) The network toplogy such as the direct blue links is the Balanced Shortest Path Tree. (b-g) the ACS execution such as: The white nodes are the leaves and the red are the scheduled nodes.

a collision-free schedule with a data latency bounded by $24D + 6\Delta + 16$ for an aggregation tree; where D is the network diameter and Δ is the maximum node degree. In the aggregation tree construction phase, the tree generated by *DAS* is like the one in NCA [23], and is constructed in a distributed way using the approach in [50]. During the data aggregation scheduling phase, in order to schedule all the nodes in a distributed way, the authors define the *node u's competitor set*. The latter is defined as the set of nodes that cannot be scheduled at the same slot as u due to messages collision. The authors proved that this set includes all the neighbors of u's parent and all the children of u's neighbors. Initially, all network nodes are in unready state. A node changes its state to ready, if and only if it is a leaf node or all its children have been scheduled. A ready node is scheduled, if and only if its *ID* is larger than all *IDs* of its ready competitor sets. After a given node is scheduled at a given time-slot, all the nodes in its competitor set label this slot as a denied slot, and delete this node from their competitor sets. An example of *DAS* execution is presented in Fig. 14. Node 8 is the first one to be scheduled because it is ready and its *ID* is larger than that all of its ready competitors (i.e., nodes 1, 2, 3 and 5 in Fig. 14(b)). After that, node 5 is chosen to be scheduled (Fig. 13(b)). A node becomes ready when all its children are scheduled, i.e., node 6 in Fig. 13(e).

The use of a sequential approach and a *connecting dominating set* structure may increase the data latency in *DAS*. Nevertheless, as the parent's time slot is higher than the ones of its children, the aggregation freshness and data accuracy are guaranteed in *DAS*.

c) ***ACS****: ACS* (Aggregation Convergecast Scheduling in Wireless Sensor Networks) is a centralized aggregation scheduling protocol. Based on the observation that the path in *Dominating Set Tree (DST)* from nodes to the base-station is not necessary optimal, the authors suggest creating a Balanced Shortest Path Tree (*BSPT*) instead of a *DST* as a data aggregation framework. Thus, the shortest path to connect each node to the base-station is used in order to minimize the number of time slots from the bottom level to the base-station. In addition, the parent-children assignments of the same depth are balanced to the largest extent in order to minimize the time latency. After the *BSPT* construction, a scheduling algorithm is started in order to schedule each node of the tree. The algorithm starts by scheduling the eligible nodes, which represent the leaf nodes that have most links with the next level. After that, the algorithm deletes the scheduled nodes from the network graph and goes to the next time slot. If there is another set of eligible nodes (leaves in new graph), the algorithm schedules them. Otherwise, the scheduling algorithm terminates. An example of *ACS* is presented in Fig. 15. The first scheduled node is 2, since it is eligible and it has more links with ineligible nodes. Next, node 1 and 3 are scheduled in the same slot because they have the same number of ineligible neighbors and they do not have a conflict at their parent nodes. As the parent node is selected only from $L-1$ level, the data latency in *ACS* can be important. Moreover, the data accuracy and aggregation freshness are ensured in *ACS* due to the fact that parent's slot is always higher than the children' ones.

d) ***PDA****:* a centralized solution, called *PDA* (Peony-tree-based Data Aggregation), with time latency $15R+\Delta-15$, has been published in [25]. By using the hop count information, network nodes are subdivided into levels, where the first level (i.e.,L_0) contains only the base-station, while the bottom one (L_R) contains only the leaf nodes. In order to create the aggregation tree, a maximal independent set (D) is selected first from the network graph G. This set is constructed in top-down manner. Initially, D contains only the base-station. Starting from the first level until the last one, for each node u in level L_i, the following test is done: If u is independent

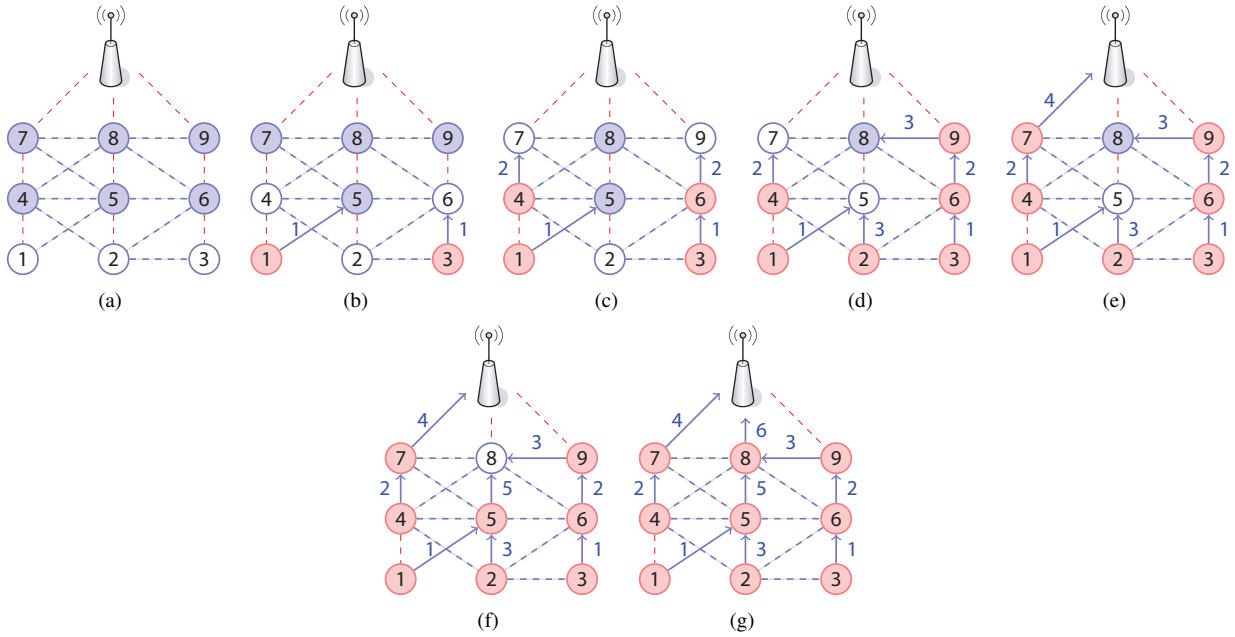


Fig. 16. *SDA* solution. (a) The network topology such as the dashed red links is the *SPT*. $(b$-$g)$ the *SDA* execution such as: The white nodes are the leaves and the red nodes are the scheduled nodes.

from all the other nodes in D, (i) u is added into D; (ii) u is removed from G; (iii) all u's neighbors are also removed from G. Otherwise, u will be removed from L_i. At the end, all the dominator nodes D will be selected. To interconnect the dominators in level L_i (i.e., D_i) with dominators in level L_{i-1} (i.e., D_{i-1}), a set of non-redundant dominates, called connectors (i.e., C), are selected. Here, a connector node x (a dominate of a dominator u) is said to not be redundant for the dominator u, if removing x will disconnect at least one of the two-hop dominator neighbors of u. The remaining nodes $(G - D \cup C)$ are denoted as dominate nodes (i.e., W). For each dominate node u in level L_i, its parent is selected from its neighbors in $D \cap L_{i-1}$. After that, the data aggregation scheduling is executed using first-fit algorithm in two steps:

1) All the dominate nodes are subdivided into node-disjoint maximum concurrent sets W_1 , W_2 , $\cdots$, W_h. A set W_i is node-disjoint maximum concurrent, if $\exists u \in W_i$, then $\nexists v \in W_i$ where u's (resp., v's) parent is neighbor to v (resp., u). Therefore, nodes in W_i can transmit data to their parents without interference in the i^{th} time slot,
2) The dominators and connectors are scheduled level by level starting from level L_R. The dominators and connectors at each level are also subdivided into node-disjoint maximum concurrent sets and then are scheduled.

The data latency in *PDA* can be important due to the fact that: (i) The leaf nodes are first scheduled; (ii) then the dominator and connector are scheduled level by level starting from the node with the highest time slot. Moreover, the aggregation freshness and data accuracy are ensured in *PDA* due to the fact that parent's time slot is higher than the children' ones.

e) ***SDA****: SDA* (Shortest Data Aggregation) [22] is a centralized heuristic that achieves time latency of $(\triangle - 1) \times R$; where $\triangle$ is the maximum node degree and R is the network radius. In *SDA*, the Shortest Path Tree *SPT* is created first. Then, the created tree is used as an input for the scheduling algorithm. Note that, *SPT* will not be used as a data aggregation framework, it will be exploited only to sort the network nodes in the scheduling process. The latter is executed for many iterations, and each iteration aims to determine a set of nodes that can be scheduled at the same time slot. The leaf nodes of *SPT* are sorted according to the number of their non-leaf neighbors, where the node with the highest number is first considered. Let S and $\overline{S}$ denote the set of sorted leaf nodes and the non-leaf nodes in *SPT*, respectively. For each node u in S, the following test will be done: If $\exists v \in \overline{S}$ where u is the exclusive neighbor of v (i.e., $\nexists w \in S - \{u\}$, in which w is neighbor to v), then u is scheduled and v is selected as its parent at this time slot. Here, u is removed from *SPT*. When node's children are scheduled, this one becomes a leaf node. When all the nodes in S are tested, the leaf nodes that are not scheduled and the new ones (in *SPT*) are considered for the next iteration, and then they are sorted and tested using the same approach. Fig. 16 illustrates *SDA* execution. Among the leaf nodes at the bottom level: 1, 2 and 3, *SDA* starts applying the test on node 2 as it has the largest number of no-leaf neighbors. Since node 2 does not have any exclusive no-leaf neighbor, it is discarded from this iteration, and nodes 1 and 3 are scheduled as they have a set of exclusive no-leaf neighbors. In Fig. 16(b), the test is applied on the new set of leaves: 2, 4 and 6, and it is repeated in Fig. 16(c) – Fig. 16(g) until all the nodes are scheduled. Besides allowing a parent selection from three levels, using the simultaneous approach helps to minimize the data latency in *SDA*. As the parent time slot is higher than the ones of its children, *SDA* ensures the aggregation freshness and data accuracy.

f) ***ISDA****: ISDA* (Improved *SDA*) [26] is a centralized solution that enhances the time latency of *SDA* [22]. This protocol generates a collision-free schedule with a data latency

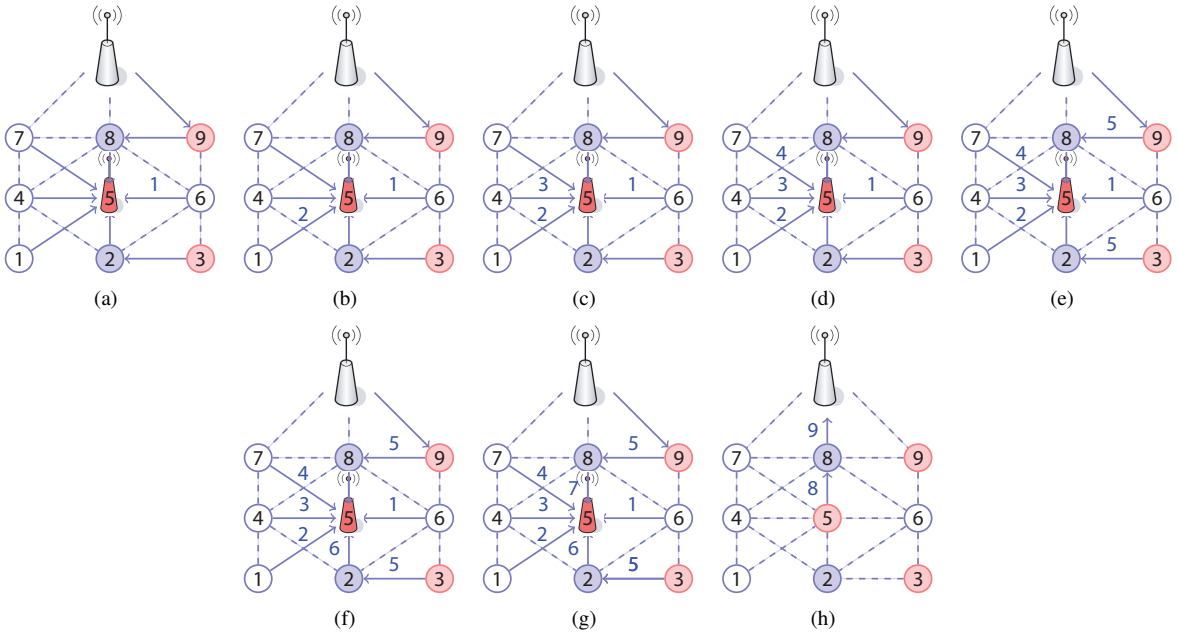


Fig. 17. *CIAS* solution. (a-i) the *CIAS* execution such as: The white nodes are the dominates, the red are the dominator nodes and the node 5 is the centralized topology.

upper-bound of $(\frac{7\Delta}{\log_2(|N|)}) \times (R-1)$ for an aggregation tree, where R is the network radius, N is the number of nodes, and Δ is the maximum node degree. *ISDA* starts by dividing the network area into equal cells. Each one of them can contain a set of nodes. In the first step, to maintain the inter-cell communication, *ISDA* creates a shortest path tree *SPT* in the graph G whose vertices are the cells. The link between the parent and its child is realized by selecting an arbitrary pair of neighboring nodes from the parent and child cell. At the end, a *SPT* is created over cells, where each cell contains a child node and each of not leaf cell contains at least one parent node. The second step is to establish the connection between each parent and the child node in each cell, using a set of nodes, to form the aggregation tree. However, there are some network nodes which may not join the aggregation tree. To overcome this problem, these nodes join the aggregation tree as leaf nodes, then will be scheduled first. To schedule the rest of nodes in the tree, the authors use the *SDA* protocol [22]. By combining *TDMA* and ensuring that parent's time slot is higher than the children' ones, *ISDA* ensures the aggregation freshness and data accuracy. Nevertheless, the data latency is affected in *ISDA* due to the scheduling process. In *ISDA* all the leaf nodes should be scheduled before the other ones.

g) ***CIAS****: CIAS* (Centralized Improved data Aggregation Scheduling) [27] is a centralized solution that aims to minimize the data aggregation latency by using *CDS* topology. The authors choose the network topology center as the aggregation tree root instead of the base-station. This choice is made to reduce the upper-bound time latency of *CIAS* to a function of the network radius R, instead of the network diameter D. The time latency in *CIAS* is at most $16R + \Delta - 14$. Here, it is important to note that Bagaa et al. in [29] have proved that the data latency upper-bound of *CIAS* is incorrect.

CIAS is carried out in two phases: (i) The Dominating Set construction; (ii) the data aggregation scheduling. The Dominating Set is constructed first using the same approach as in [50]. Then, the data aggregation scheduling is done in two successive steps. In the first one, the data are aggregated from all the dominates to the dominators by scheduling the maximum number of tuples (dominate, dominator) simultaneously at each time slot. In the second one, the dominators aggregate their data to the root node, level by level, using a non-redundant set of dominates called connectors. The latter are selected to connect each dominator to all dominators in lower level and are within two-hop away. An example of *CIAS* is presented in Fig. 17. Before starting the scheduling process, the topology center (i.e., node 5) is selected as a new base-station. Then, the data aggregation tree rooted to the new base-station is constructed using DS (Dominating Set). In the aggregation scheduling step, all the dominate nodes (i.e., nodes 1, 3, 4, 6 and 7) are scheduled first. Afterwards, the dominator nodes are scheduled (nodes 2 and 8) level by level. At the end, the path from node 5 to the real base-station is scheduled. After aggregating the data by the topology center, the latter resends the aggregated value to the real base-station by using the shortest path. However, this technique may increase the data latency in *CIAS*. Due to the fact that message collision is avoided (CIAS uses *TDMA*) and the parent's time slot is always higher than the children' ones, the data accuracy and aggregation freshness are ensured in *CIAS* solution.

h) ***SAS****: SAS* (Sequential Aggregation Scheduling) [28] is a centralized solution. This solution generates a collision-free schedule with a latency bound of $15R+\Delta-4$; where R is the network radius and Δ is the maximum node degree. Bagaa et al. in [29] have proved that the data latency upper-bound of *SAS* is incorrect.

SAS is executed in two steps: the aggregation tree construction and the data aggregation scheduling. The first one is constructed using an existing approach [50]. The second one schedules all the dominate nodes using the same approach

as [27]. To forward the aggregated data from the dominators to the base-station, the network is divided into levels such that the nodes in even (resp., odd) levels are the dominators (resp., non-redundant connectors). The dominators and the connectors are scheduled level per level, from the bottom to the base-station as follows:

1) For each dominator (resp., non-redundant connector) u in level i, its parent $p(u)$ is selected from level $i-1$ as non-redundant connector (resp., dominator) whose *ID* is the smallest one. At the end, two sets will be generated; the first one (i.e., N_i) is constituted with dominators (resp., non-redundant connectors) in level i, whereas the second one (i.e., P_i) is constituted with the parents of N_i.
2) The set of links from the nodes in N_i to the ones in P_i is denoted by A_i.
3) The conflict graph of A_i, denoted by $CG(A_i) = (V, E)$, is generated. V is the set of links $(u, p(u))$ in A_i and E is the set of conflicting edges between them. Here, there is an edge between $(u, p(u)) \in V$ and $(v, p(v)) \in V$, if and only if there is a conflict between them i.e., $(u, p(v)) \in A_i$ or $(v, p(u)) \in A_i$.
4) The first-fit coloring of $CG(A_i)$ is computed. Then, each link in A_i with color j is scheduled in the j^{th} time-slot of the i^{th} level.

In *SAS*, the use of *CDS* structure may increase the data latency. Besides using *TDMA* schedule, the fact that the parent's time slot is greater than the children' ones allow SAS to ensure the data accuracy and aggregation freshness.

i) ***DAS-ST****: DAS-ST* (Data Aggregation Topology based on Semi-structured Topology) [29] is a centralized solution. *DAS-ST* uses *MIS* (Maximal Independent Set) to reduce the data aggregation latency. The data latency upper-bound achieved by DAS-ST is $(\lfloor \frac{2\pi}{arccos(\frac{1}{1+\epsilon})} \rfloor + 4)R + \Delta - 4$, where R is the network radius, Δ is the maximum node degree, and $0.05 < \epsilon \leq 1$. *DAS-ST* starts by dividing the network into levels with respect to the hop-count information. At each level, the nodes in *MIS* are colored as black nodes. Nodes in *MIS* at level L must do not have links with other ones in *MIS* at level $L-1$ or $L+1$. The remaining nodes are colored as white nodes. In *DAS-ST* each black (resp., white) node should select its parent only from white (resp., black) neighbors. In order to minimize the data aggregation latency, *DAS-ST* is based on two key design features, which are: (i) simultaneous execution of aggregation tree construction and scheduling; (ii) selecting parent node from three levels and then maximizing the choices of parents for each node, which maximize the time slot reuse. The network nodes in *DAS-ST* are scheduled level by level starting from the bottom one. If a node u in level L gets scheduled, it selects the minimum available time slot TS_u that avoids collisions. Then, according to TS_u, node u selects its best candidate parent w from $L+1$, L, and $L-1$. The best candidate parent w must have the minimum number of unscheduled neighbors, in order to maximize the time slot reuse in the neighboring nodes. Selecting w as the best candidate parent for node u does not mean that this selection is approved right away. Indeed, a better choice could exist when another node v leads to more optimal solution, if it is scheduled before u. If the algorithm starts scheduling v, v might select z as a parent at $TS_v = TS_u$; where w (resp.,z) is within the transmission range of v (resp., u). In this case, a collision will occur if the two nodes u and v are scheduled at the same time. To resolve this conflict, one of them must be scheduled before the other one. If the first one, e.g., u, is scheduled at TS_u, then the second one, e.g., v, must be scheduled at time slot TS_v higher than TS_u. To allow a large number of nodes to reuse the smallest time slots, the node whose candidate parent has the minimum number of unscheduled neighbors is scheduled first. If many candidates parents have the same number of unscheduled neighbors, the child node with the minimum number of unscheduled neighbors will be scheduled first. Note that, *DAS-ST* allows simultaneous execution of aggregation tree construction and scheduling. In addition to selecting the parents from three levels, the parent can also be selected from the already scheduled nodes, which represent a novel point compared to the other solutions. Thus, the data latency in *DAS-ST* is significantly reduced. Moreover, the aggregation freshness and data accuracy are ensured due to the fact that parent's slot is higher than the children' ones.

j) ***DAS-UT****: DAS-UT* (Data Aggregation Topology based on Unstructured Topology) [29] is a centralized solution that aims to minimize the data aggregation latency. *DAS-UT* is an enhanced version of *DAS-ST*. DAS-UT relaxes more the rules to select parents by excluding the use of the MIS. *DAS-UT* allows each node u to select its candidate parents from its neighborhood, which belong to the three levels without restriction. For the same reason as *DAS-ST*, the data latency is minimized in *DAS-UT*. Furthermore, as the parent's time slot is higher than the children' ones, the *DAS-UT* ensures the aggregation freshness and data accuracy.

2) ***TDMA and transmission power control based techniques****:*

a) ***JSPC****:* Incel et al. present a new centralized solution, called *JSPC* (Joint Scheduling and Power Control) in [30], to minimize data aggregation latency by combining *TDMA* and power transmission control mechanisms. Initially, a data aggregation tree with *B-rule* is created. Afterwards, the data aggregation scheduling is executed in two phases: (i) time slots distribution over network nodes; (ii) the power transmission adjustment of each node. In the first phase, assuming a protocol interference model, a set of slots are distributed via the data aggregation tree, where the parent slot can be lower than the child one. Accordingly, *JSPC* does not ensure the aggregation freshness, and hence the data of different frames in the monitoring application can be aggregated together. In the second phase, the power transmission of each node is adjusted in order to support the more realistic *SINR* model.

As the child's slot can be higher than the parent's one, the data accuracy and aggregation freshness are significantly decreased, and the data latency is minimized.

B. Hybrid-multiplexing

1) ***TDMA and CDMA based techniques****:*

a) ***LEA****: LEA* (A low-latency and energy efficient algorithm for convergecast in wireless sensor networks) [31]

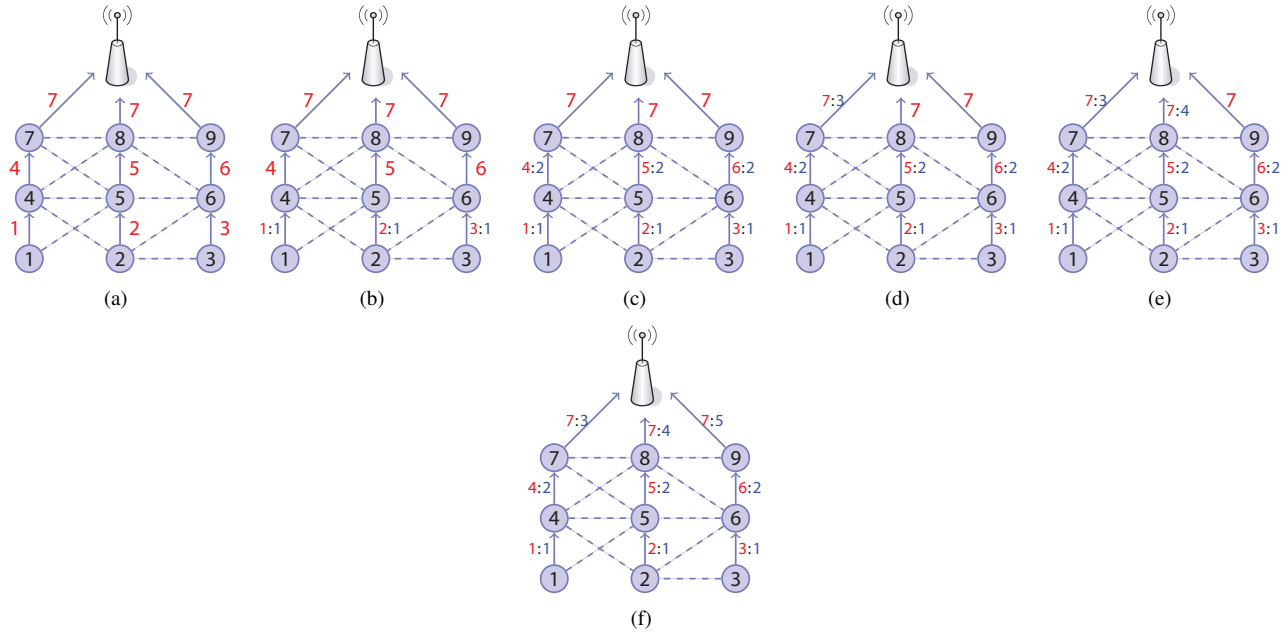


Fig. 18. LEA solution. (a-f) the *LEA* execution such as: The red numbers on edges are the codes and the blue numbers on edges are the time slots.

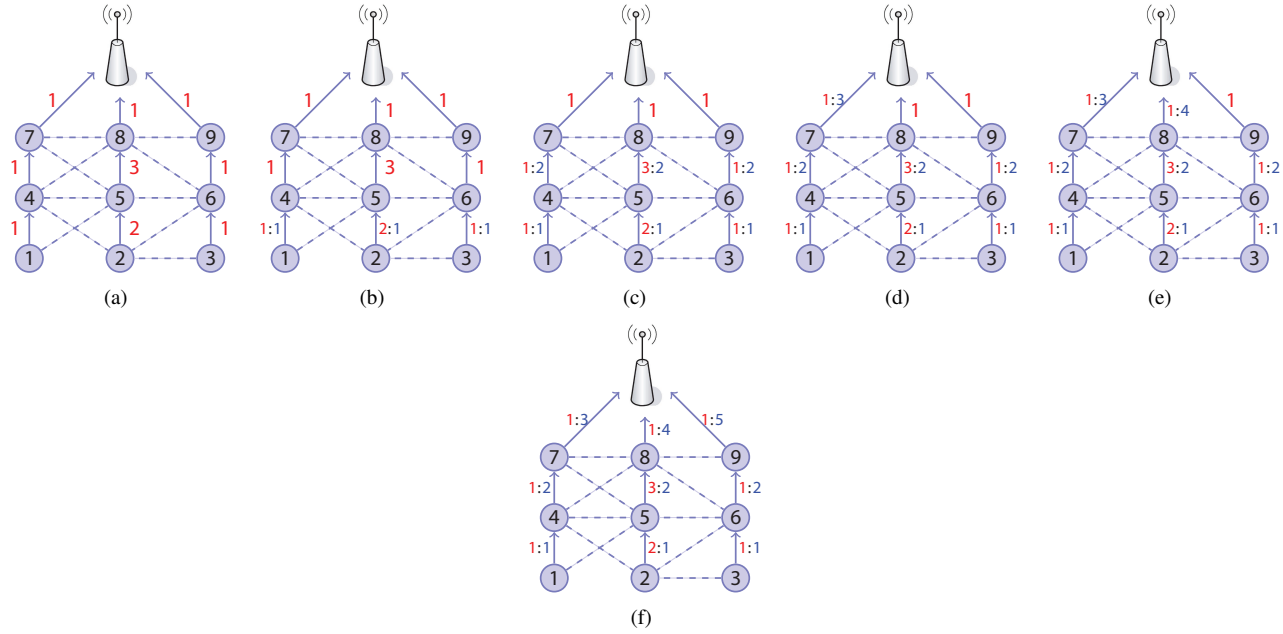


Fig. 19. *RBCA* solution. (a-f) the *RBCA* execution such as: The red numbers on edges are the frequencies and the blue numbers on edges are the time slots.

is a centralized solution that uses both *TDMA* and *CDMA* mechanisms. Here, *TDMA* mechanism is used to synchronize the communication between the children and their parents in the aggregation tree. Each child node has an assigned slot to transmit its data to the parent node. To overcome the problem of collision, two children of the same parent must use two different slots. The parent aggregates and sends the received data to its parent by using a time slot higher than all children' slots. Here, the number of children has a direct impact on data aggregation latency. In order to minimize the data latency, a data aggregation tree with *B-rule* is constructed, which means that each node cannot get more than *B* children in the tree. Therefore, multiple transmissions can be done simultaneously without collision. However, a collision can occur if two nodes u and v have (i) different parents and (ii) u's parent or v's parent is within the transmission range of v or u, respectively. To address this limitation, the authors propose to use *CDMA* mechanism in which u and v use two different codes. Each node in the network uses two kinds of codes, one for transmission and another one for reception. The children of the same node use the same transmission code which is used as the reception code by their parent node. Fig. 18 shows an example of *LEA* execution. After the creation of *B-rule* tree, the *CDMA* codes are distributed among network nodes (Fig. 18(a)). After that, all the network nodes are scheduled starting from the bottom level (Fig. 18(b) – Fig. 18 (f)).

As the parent's slot is higher than the ones of its children, *LEA* ensures the aggregation freshness and data accuracy. Furthermore, the time latency can be minimized due to the use of *B-rule* tree.

b) ***DAC****: DAC* (Spanning tree based algorithms for low latency and energy efficient data aggregation enhanced convergecast (DAC) in wireless sensor networks) [32] is a centralized solution that uses *TDMA* and *CDMA* mechanisms to save energy and minimize data latency. By controlling the length of a slot at each level, *DAC* can support any kind of aggregation function (i.e., perfect and partial). To ensure the aggregation freshness, the parent's time slot must be higher than all the children's ones. Moreover, the length of a time slot is identical for all the network nodes, if a perfect aggregation function is used. Otherwise, the length of a time slot at level i is considered as the delay to transmit the largest size data packet from any node in level i to its parent in level $i-1$. By using the interference protocol model, the authors assume that the network could be represented with a graph noted G. Such as, there is an edge between two nodes if and only if both nodes are within the transmission range of each other. Besides, a weight $w(i,j)$ is associated with an edge (i,j) indicating the cost in terms of energy consumption between two vertices i and j. The weight of an edge can be decreased or increased according to the power transmission used by the source node. The power transmission of each node is adjusted according to its energy and its distance from the destination node.

Based on the observation that for each type of aggregation function there is an appropriate tree structure leading to minimize the energy consumption, three algorithms have been proposed to create the aggregation tree structure.

1) *SPT* (Shortest-Path spanning Tree): If the aggregation process is done at the base station (i.e., MEDIAN and MODE), then the number of packets from the leaves to the base-station cannot be reduced. The best way to minimize the energy consumption is by reducing the number of hops from each leaf to the base-station. Therefore, the *SPT* is the best solution that can be used if the aggregation process is done at the base-station,
2) *MST* (Minimum Spanning Tree): If the aggregation function is perfect, then the size of the aggregation result is one. Therefore, to save energy consumption each node selects its parent as the closest node (i.e., the edge which has the smallest weight),
3) *SPT-MST*: *SPT-MST* is a new structure that can support any kind of aggregation functions. The *SPT-MST* inherits the advantage of the two structures; that is, identical to *SPT* if the aggregation process done at the base-station and identical to *MST* if the aggregation function is perfect.

According to the aggregation function type, *DAC* uses one of the above trees as data aggregation tree (*DAT*). In order to reduce the data latency a refinement step is applied on *DAT*. If a node has C children such that C is more than a predefined threshold B, then $C-B$ children would be distributed among the other parents. Therefore, the *DAT* becomes a *B-rule* tree. At the end, the nodes in *DAT* are scheduled using *TDMA* and *CDMA* techniques, like in *LEA*. By combining the use of *B-rule* tree as well as *TDMA* and *CDMA* techniques, *DAC* significantly reduces the data latency. Moreover, *DAC* ensures the aggregation freshness and data accuracy, since the parent's slot is higher than the children' ones.

2) ***TDMA and FDMA based techniques****:* The miniature hardware design of nodes in *WSNs* does not permit employing complex radio transceivers required for spread spectrum codes. Thus, the use of CDMA technique is a complex process and is not energy efficient for wireless sensor networks. Therefore, *FDMA* technique is proposed instead. The *FDMA* use in *WSNs* provides hybrid multiplexing when scheduling network nodes, and hence reduces further the data latency. Based on the observation that the recent transceivers in *WSN*, like *CC2420* [18], provide multiple frequencies, such as 16 orthogonal frequencies with *5MHz* spacing, two solutions, called *RBCA* (Receiver-Based Channel Assignment) and *MCSA* (Multi-Channel Scheduling Algorithms for fast aggregated convergecast in sensor networks) have been recently published in literature. Based on the observation that a node's transceiver cannot hear simultaneously more than one message, the children of the same parent should use different slots. Therefore, assigning different channels to the children of the same parent does not reduce the data latency. When a parent receives over different channels, a predefined time-slot should be used to allow the channels switching, which increases the time latency. Thus, in these solutions each node uses at most two channels: one is used to receive data from its children nodes and the other to transmit the aggregate result to its parent.

a) ***RBCA****: RBCA* (Receiver-Based Channel Assignment) [30], [33] is a centralized solution that uses *B-rule* tree. *RBCA* takes into account the new transceivers capacity to minimize data aggregation latency by using both *TDMA* and *FDMA* mechanisms. *RBCA* assumes that a *B-rule* tree is already created, in order to be used as a data aggregation framework. Note that, the parents of all the nodes in *B-rule* tree are selected only from their next level. In the scheduling step, all nodes in the network initially use the same channel, which is called the default channel. Afterwards, a set of channels S are distributed over network nodes to eliminate the interferences between nodes, and then reduce the number of slots that would be used later to schedule network nodes. φ is initialized to the set of all parent nodes (non-leaf nodes) in *B-rule* tree. For each parent node w in φ, *RBCA* assigns a receiver channel c from S, which is used as a sender one for w's children. To eliminate the interferences at w, c should not be assigned to another parent v whose child creates an interference at w. In order to enhance the simultaneous transmissions, φ is sorted in a way that the parent node which has the highest conflicting with other nodes is first considered. Here, the interfering link of a parent node w in level L is defined as the maximum number of collisions that can be created on w, if all the nodes in previous level (level $L+1$) are scheduled. For each parent w in φ, the following test is done: If there is an available channel c in S that can be assigned to w without creating an interference with nodes in γ (i.e., nodes that are already assigned a receiver-channel), then (i) c is assigned to w; (ii) w is put into γ. Otherwise, w remains on the default channel and the interference conflicts have to be solved using *TDMA* mechanism. An example of *RBCA* is depicted in Fig. 19,

where the maximum number of channels that can be used is 3. Note that, the aggregation freshness and data accuracy are not ensured in *RBCA*, since the parent's time slot can be lower than the child's one. Nevertheless, the data latency can be significantly minimized, due to: (i) the use of *FDMA* technique; (ii) the parent's time slot can be lower than the child's one.

*b) **MCSA**:* MCSA [34] is a centralized solution that aims to maximize the data rate without taking into account the aggregation freshness. This is done by using the *TDMA* mechanism and *FDMA* channels. Based on the assumption that the routing tree is already created, *MCSA* starts the scheduling process by dividing the network nodes into square grids. The nodes at each grid are sorted according to the decreasing number of their children in the aggregation tree. After that, two successive steps must be accomplished to minimize the data latency, which are the assignment of channels and time slots. Here, the channel assignment aims to minimize the number of nodes transmitting on the same channel. Therefore, the channels are distributed without any conflict on the nodes for each grid, starting from the node that has the maximum number of children. As the aggregation freshness is not important in this case, the time slots can be reused across separate grids. The aggregation freshness and data accuracy are not ensured in *MCSA*, since child's slot can be higher than the parent's one. Meanwhile, the data latency is reduced in *MCSA*, since the parent's slot can be lower than the one of its child and the use of multi-channels technique.

C. Discussion of existing solutions

Fig. 20 depicts the taxonomy of slotted data aggregation scheduling protocols. It also gives a comprehensive comparison among the existing solutions in terms of data aggregation goals, algorithm nature, parent level, topology, interference model, application nature, aggregation function and data latency upper-bound. All slotted-based data aggregation scheduling protocols, except *DAS*, are centralized. The protocols of this category are based on the cross-layer principle (routing and MAC layers). They are resilient to clock drifts, as long as we assume that nodes have synchronized clocks. Most of the existing solutions in this category: (i) save energy consumption (nodes are mostly in sleep mode); (ii) avoid collisions due to the use of *TDMA* mechanism; (iii) use tree structure as routing scheme. Moreover, they can be used in either event-driven or monitoring applications. As mentioned before, there are two classes of slotted data aggregation scheduling protocols: temporal-multiplexing and hybrid-multiplexing. Each class is further subdivided into two subclasses.

1) **Temporal-multiplexing:**
   a) **TDMA-based:** Many solutions [23], [12], [24], [25], [22], [26], [27], [28], [29], employ this technique. All these solutions use only the interference protocol model, and ensure data accuracy and aggregation freshness. To reduce the data latency many techniques are used in these solutions, where the most efficient are those which: (i) allow parents selection from three layers and (ii) execute tree construction and network nodes scheduling simultaneously.
   b) **TDMA & Transmission power control:** There is only one solution *JSPC* [30] in this category, which uses both the protocol and *SINR* interference models. The aggregation freshness and data accuracy are affected in *JSPC* due to the fact that parent's slot can be lower than the child's one.
2) **Hybrid-multiplexing:**
   a) **TDMA & CDMA:** In this class, there is two protocols *LEA* [31] and *DAC* [32] which are sequential. Furthermore, they ensure data accuracy and aggregation freshness. Solutions in this category assume the interference protocol model. In contrast to all slotted data aggregation scheduling solutions, which uses perfect aggregation function, *DAC* uses any kind of aggregation functions. The main drawback of existing solutions in this category is in the fact that *CDMA* cannot be employed in existing sensors.
   b) **TDMA & FDMA:** *MCSA* and *RBCA* combine *FDMA* and *TDMA* in order to enhance the simultaneity of transmissions and hence reduce the time latency. Both solutions support both protocol and *SINR* interference model. In this category of protocols, the parent's slot can be lower than the child one, which significantly affect the aggregation freshness and accuracy.

D. Theoretical optimization of data latency mechanisms

As depicted in Fig. 20, the aim of slotted data aggregation scheduling solutions is the reduction of data latency. For this reason, existing solutions in this category uses a cross-layer approach that combines the formation of the aggregation tree with medium access scheduling. A node in a distributed solution has only a local view about the network and thus the selection of the parent and the set of time-slot could not be optimal. Based on this observation and in order to reduce the data latency, most of existing solutions in the literature adopt a centralized approach. To the best of our knowledge, DAS is the unique solution which is distributed in the literature. In the the centralized solutions ([23], [24], [25], [22], [26], [27], [28], [29], [30], [31], [32]), whenever a topology change occurs, the base station has to gather information about the new network topology, recompute the schedule, and disseminate it to the whole network. As the topology might frequently change, the centralized approach incurs high message overhead, high energy consumption, and low scalability. On the other hand, the distributed approach assumes that each topology change will affect a small part of the network, and only the nodes of this part have to recompute their schedule in a distributed way, which leads to high scalability, low energy consumption and fast adaptability to network changes.

As mentioned in subsection III-B4, each node in the network has a direct link with neighbors belonging to the following three levels: next-level, same-level and previous-level. In order to reduce the data latency, existing solutions aim to select for each node the parent which reduces the number of

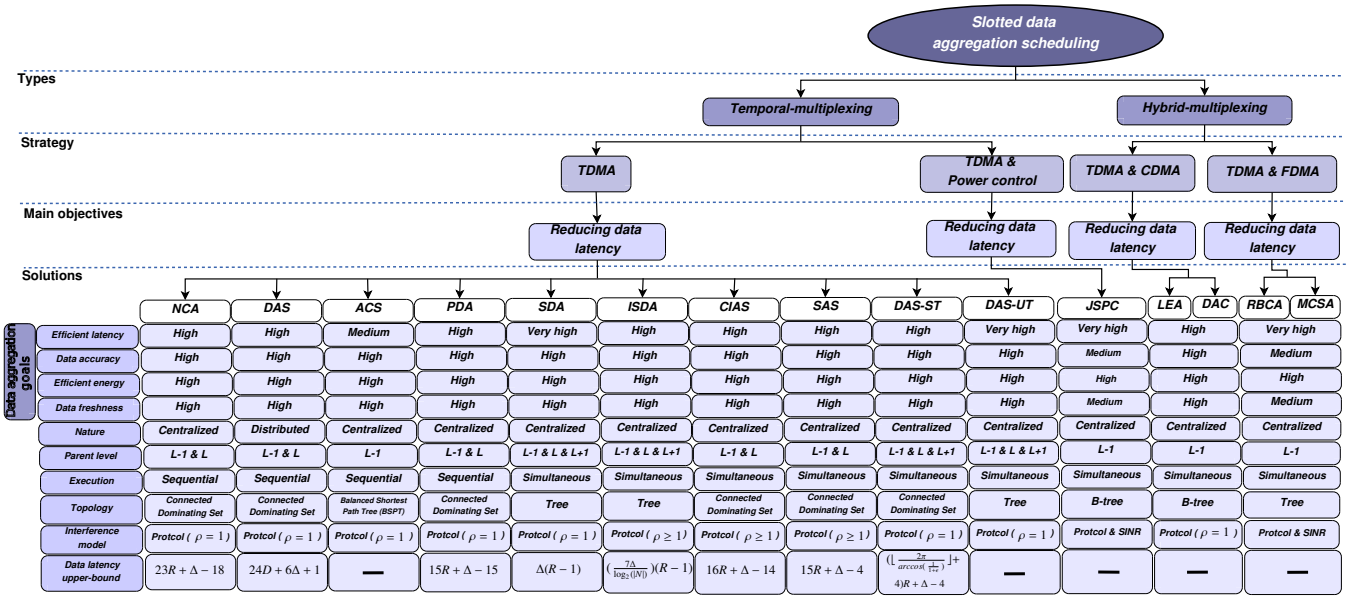


Fig. 20. Slotted-based data aggregation scheduling protocols taxonomy

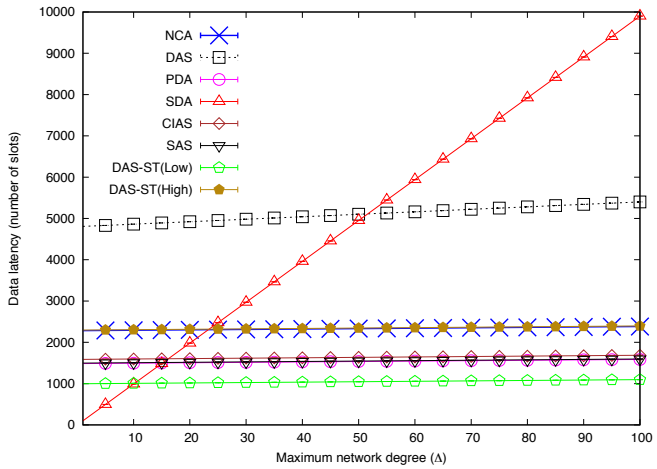


Fig. 21. Data latency upper-bound of exiting solutions

conflicts in the network and thus increases the time slots reuse. For this reason, most of existing solutions aim to increase as much as possible the number of candidate parents for each node in the network when constructing the aggregation tree. NCA [23], DAS [12], PDA [25], CIAS [27] and SAS [28] solutions increase the number of candidate parents by allowing each node to select its parent from the same-level and the next-level. Meanwhile, SDA [22], ISDA [26], DAS-ST [29] and DAS-UT [29] aim to increase the number of candidate parents set by allowing each node to select its parent from the next-level, the same-level and the previous-level. The formation of the aggregation tree and medium access scheduling process are intertwined or executed one after the other. As depicted in subsection III-B4, the data latency would be reduced when the two processes are intertwined. In order to reduce more the time latency, SDA [22], ISDA [26], CIAS [27], SAS [28], DAS-ST [29] and DAS-UT [29] intertwine the formation of tree and the transmission scheduling.

In order to demonstrate the superiority of TDMA-based solutions in terms of data latency, authors of existing works striven to upper-bound the data latency of each protocol. Usually, graph theory, graph coloring and theoretical geometry are used to formulate the upper-bound of data latency. The data latency upper-bound is formulated using Δ and R, where Δ is the maximum degree and R is the network radius. Based on the assumption that both Δ and R are on the same order of the network size, existing solutions aim to reach bounds where Δ and R are additive factors and avoid to reach bounds where Δ and R are multiplicative factors. SDA [22] is the first solution which upper-bounded its data latency by $(\Delta-1)\times R$. In this case, Δ and R are multiplicative factors. ISDA [26] achieves data latency upper-bound which equals to $(\frac{7\Delta}{\log_2(|N|)})\times(R-1)$, where N is the number of nodes, and hence enhances the data latency upper-bound of SDA [22]. However, Δ and R remain multiplicative factors in the data latency upper-bound in ISDA [26]. DAS [12] enhances the data latency upper-bound of SDA [22] and ISDA [26] by making Δ and R as additive factors. The time latency of DAS [12] is $48R+6\Delta+1$. The data latency upper-bound of NCA [23] is $23R+\Delta-18$ which enhances the data latency of DAS [12]. A significant enhancement of data latency upper-bound is proposed by CIAS [27], SAS [28] and PDA [25]. The data latency upper-bound of CIAS [27], SAS [28] and PDA [25], are $16R+\Delta-14$, $15R+\Delta-4$ and $15R+\Delta-15$, respectively. The data latency achieved by DAS-ST [29] is $(\lfloor\frac{2\pi}{arccos(\frac{1}{1+\epsilon})}\rfloor+4)R+\Delta-4$, where $0.05<\epsilon\leq 1$. According to the value of ϵ, DAS-ST [29] achieves a data latency upper-bound between $10R+\Delta-4$ and $23R+\Delta-4$ which is an enhancement of the one achieved by NCA. Fig. 21 shows the data latency upper-bound. In Fig. 21, Δ is varied and R is fixed to 100. The first observation we can make from the figure is that the increase of Δ has a negative

impact on the upper-bound data latency of existing solutions. We notice that DAS-ST outperforms the other solutions. Since Δ and R are additive factors in DAS, NCA, PDA, SAS and CIAS, these solutions outperform SDA in terms of data latency upper-bound. DAS scales better than SDA but remains less efficient compared to the other solutions. Nevertheless, even SDA seems to be not performant with large degree networks, it performs very well with small degree networks ($\Delta < 10$).

VI. General discussion and future directions

In this section, we summarize the different techniques used in each class of data aggregation scheduling protocols and highlight the impact of these techniques on data aggregation design goals. Open problems and future directions are presented and discussed.

Fig. 22 shows how the reviewed unslotted data aggregation scheduling solutions tackle the different data aggregation objectives. As noticed in section IV, the use of aggregation function has an impact on all data aggregation goals. Moreover, the optimal distribution of *WT* among network nodes has an impact on collision, aggregation freshness, and data accuracy. In contrast, minimizing the *WT* length has an impact on saving energy and data delay. Fig. 23 depicts the impact of slotted data aggregation scheduling on design goals. As shown in section V, reducing energy consumption and avoiding packets collision are achieved by using both time division into slots and aggregation function. The data aggregation mechanism minimizes packets collision and saves energy consumption by reducing the number of packets transmitted by each node. The use of time-slot allows nodes to switch off theirs transceivers most of the time (i.e., idle time) and avoid simultaneous transmissions by multiple nodes. Recall that data accuracy and aggregation freshness in slotted data aggregation scheduling are better due to: (i) the use of aggregation function which reduces the number of transmitted packets, and hence allows the base-station to receive the aggregated data from whole network nodes within each frame; (ii) the parent's slot is greater than the children ones. The data aggregation mechanism reduces the number of packets in the network, which helps to receive all aggregate values in the same frame. Meanwhile, when the parent's time slot is greater than children' ones, the parent can receive children data before forwarding the aggregate result. As noticed in section III, the data delay is influenced by two parameters: (i) time slot distribution; (ii) data aggregation structure.

Fig. 24 draws a clear timeline of the different data aggregation scheduling protocols reviewed in this paper. This timeline may help the reader to clearly understand the trend development for data aggregation scheduling protocols. Recall that the hybrid approach of unslotted data aggregation scheduling achieves a best distribution of *WT*, and hence achieves a best performance in terms of data latency, accuracy and aggregation freshness compared to the other approach. For this reason, as shown in Fig. 24, the hybrid approach is the subclass which attracts more attention than the other recent ones. The above Figure shows that *TDMA* and *TDMA & FDMA* classes attracted more attention in recent years, due to the fact that both mechanisms can be used in exiting motes, and can reduce the time latency efficiently.

We noticed in section V that the slotted data aggregation scheduling protocols offer better performance in comparison to the unslotted ones. The slotted-based protocols enhance the unslotted-based ones in terms of energy consumption, data latency, data accuracy, avoidance of collisions and aggregation freshness. Moreover, the slotted-based data aggregation scheduling protocols can be used in all kinds of applications (monitoring and event-driven) in contrast to the unslotted-based ones which can only be used in monitoring applications. The main limitation of unslotted-based solutions is related to not considering message collisions and communication delays, whereas the limitations of slotted-based solutions are: (i) the need for a global synchronization due to the use of *TDMA* mechanism; (ii) they have never implemented or tested on real sensors; (iii) they do not consider network dynamics such as adding new nodes to the network, or the failure of nodes or links. The advantages of unslotted data aggregation scheduling protocols compared to the slotted-based ones are: (i) unslotted are not limited by the strong assumption that the whole network must be synchronized; (ii) most of unslotted-based protocols are distributed.

As we have discussed before, the data aggregation scheduling protocols reduce the data latency efficiently and ensures the aggregation freshness. However, some limitations still remain in the existing solutions, which need to be addressed. Based on these issues, we will draw the future directions to be considered in data aggregation scheduling protocols. Most of the existing solutions rely on strong assumptions on the aggregation function which limits their use. They assume that the aggregation function is perfect. However, recent applications such as security surveillance [3], use complex aggregation functions. One prominent direction is the design of aggregation scheduling solutions that support less restrictive aggregation functions in order to enlarge their applicability in actual applications.

Future works in unslotted-based class, will likely focus on hybrid upstream-downstream dependent *WT*, since it offers better performance in terms of data latency and aggregation freshness. Best results can be achieved, if the existing solutions, in addition to the aggregation delay, take into account the communications delay and messages collisions.

The main limitation of slotted-based data aggregation scheduling protocols is in the fact that they are centralized and not implemented or tested in real sensors. The future works in this class should be distributed. Moreover, a real deployment of proposed solutions over sensors is needed. In order to overcome the constraint that all network nodes should be synchronized, a local synchronization can be used such as the one used in *DOZER* [51]. The harsh conditions that the nodes operate in, and the limited on-board energy supply make them susceptible to failure. Moreover, the wireless links often suffer from interferences and hence packets can be lost before reaching the base-station. Thus, failures of links and nodes become an inevitable issue, which can dramatically reduce the data throughput and make the communication infrastructure unusable. Further, the slotted-based protocols should be scalable for larger number of nodes and should consider network redeployment when new nodes join the network. In order to overcome this limitation the following

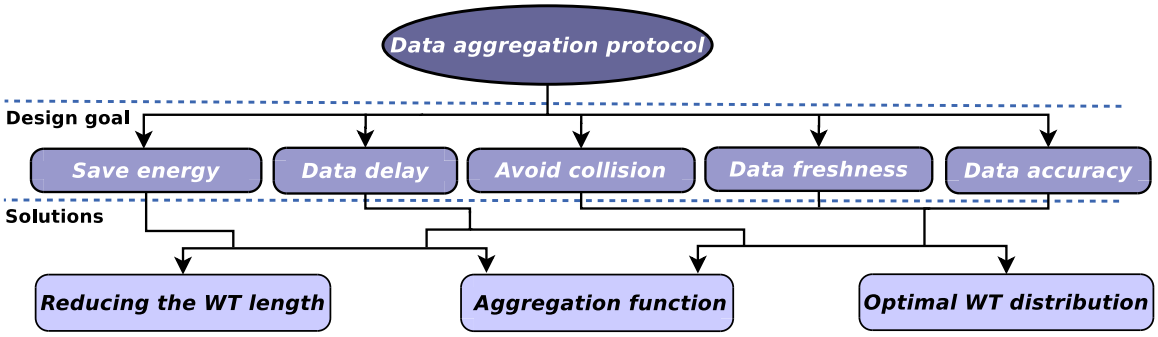


Fig. 22. Impact of unslotted data aggregation scheduling on design goals

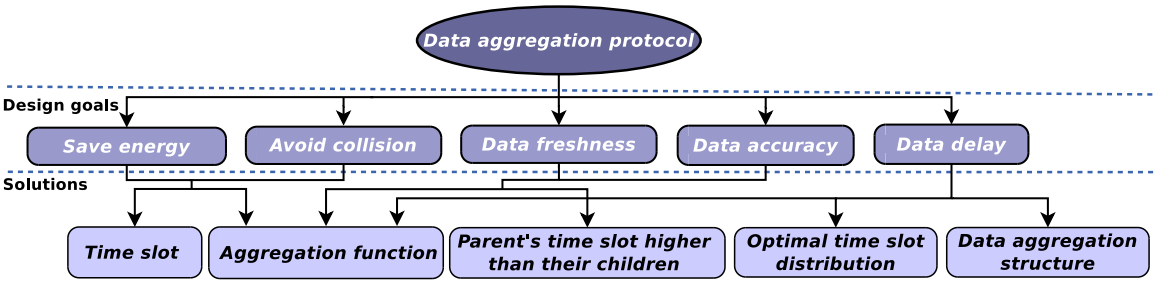


Fig. 23. Impact of slotted data aggregation scheduling on design goals

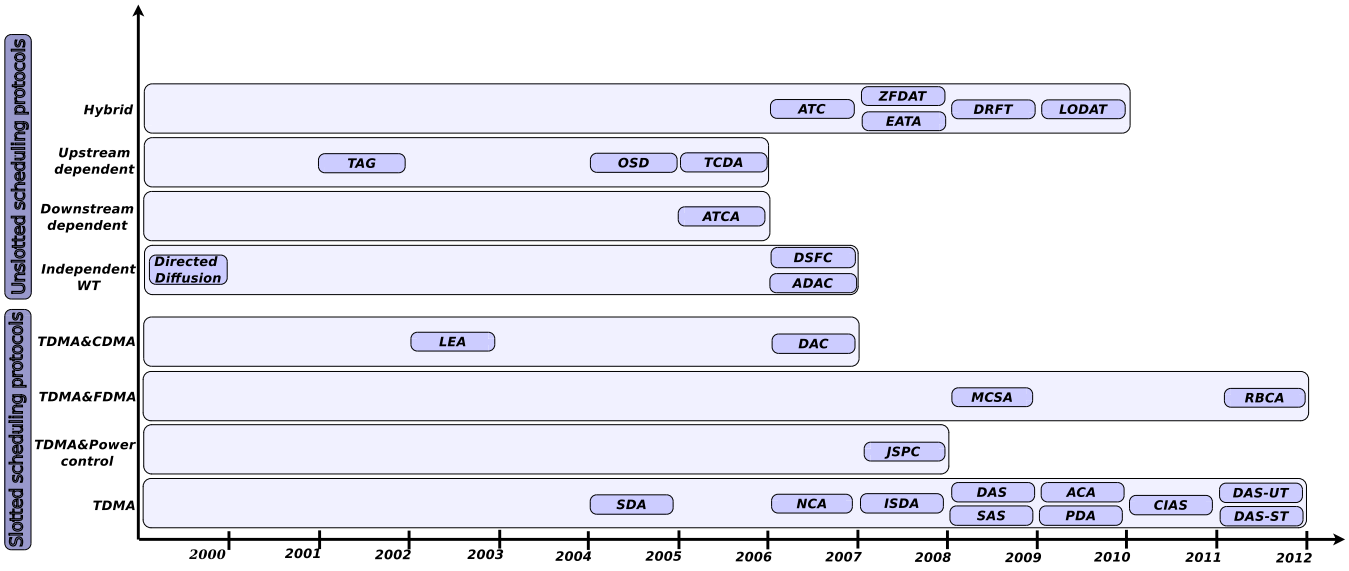


Fig. 24. Timeline of data aggregation scheduling protocols

techniques may be used:

1) Selecting opportunistic links for sensor nodes; each node selects its parent, which not only reduce data latency but also has a good link quality. A recent solution which explores this technique is published in [52].
2) The use of multi-path structure instead of a tree structure. The use of a multi-path structure allows nodes to select another parent when a link and/or node failure occurs. Recently, new solutions have been published in [11], [53] and [54] based on this technique.
   A prominent direction, would be the design of fault tolerant data aggregation scheduling. Such solutions could rely on techniques used in solving scheduling with uncertainty problems such as securing the data aggregation scheduling protocols against some attacks and keeping minimum-latency aggregation schedule under topological changes.

All solutions, which use either *TDMA* & Transmission power control or *TDMA* & *FDMA* are sequential. That is, a tree structure is created then the network nodes are scheduled. One open direction in this classes is to explore the cross-layer advantage by using a simultaneous approach. A new solution which intertwines the tree construction, and time-slots and channels allocation, has been recently published in [55].

VII. Conclusion

Data aggregation scheduling is currently an important topic for *WSN* applications, which require that the aggregated

data reach the base-station in a bounded delay. This paper investigated the data aggregation principle and focused on features and requirements that should be implemented by data aggregation scheduling protocols. In particular, the paper reviewed existing solutions by proposing a clear taxonomy to classify these works. The surveyed solutions aim to reduce the data latency, increase the data accuracy and aggregation freshness with a good distribution of nodes' waiting time (*WT*). According to this one, we have classified existing solutions into two classes: the first one is unslotted data aggregation scheduling and works at routing layer; whereas the second one is slotted data aggregation scheduling and relies on a cross-layer design (Routing and *MAC*). Unlike the unslotted-based algorithms which consider only the time of data aggregation process when distributing *WT* over network nodes, slotted-based solutions consider the time of both data aggregation process and communications. For this reason, in recent years the slotted-based solutions attracted more attention than the unslotted-based ones. Finally, we shed some light on new directions and open issues. We showed that each class has some limitations that need to be considered in future works.

REFERENCES

[1] A. Somov, I. Minakov, A. Simalatsar, G. Fontana, and R. Passerone, "A methodology for power consumption evaluation of wireless sensor networks," in *Proc. IEEE ETFA'09*, September 2009, pp. 1–8.

[2] M. Hefeeda and M. Bagheri, "Forest fire modeling and early detection using wireless sensor networks," *Ad Hoc & Sensor Wireless Networks*, vol. 7, no. 3-4, 2009.

[3] D. Dudek, C. Haas, A. Kuntz, M. Zitterbart, D. Krger, P. Rothenpieler, D. Pfisterer, and S. Fischer, "A wireless sensor network for border surveillance," in *Proc. ACM SenSys'09*, November 2009, pp. 303–304.

[4] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita, "A line in the sand: A wireless sensor network for target detection, classification, and tracking," *Elsevier Computer Networks*, vol. 46, pp. 605–634, 2004.

[5] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network aggregation techniques for wireless sensor networks: a survey," *IEEE Wireless Commun.*, vol. 14, no. 2, pp. 70–87, 2007.

[6] K. Akkaya and I. Ari, *Handbook of Computer Networks: LANs, MANs, WANs, the Internet, and Global, Cellular, and Wireless Networks*, 2007, vol. 2, ch. In-Network Data Aggregation in Wireless Sensor Networks, pp. 1131–1146.

[7] R. Rajagopalan and P. K. Varshney, "Data-aggregation techniques in sensor networks: A survey," *IEEE Commun. Surveys & Tutorials*, vol. 8, no. 1-4, pp. 48–63, 2006.

[8] C. Chen, B. Tian, Y. Li, and Q. Yao, "Data aggregation technologies of wireless multimedia sensor networks: A survey," in *Proc. IEEE ICVES'10*, July 2010, pp. 83–88.

[9] A. Bachir, M. Dohler, T. Watteyne, and K. Leung, "MAC essentials for wireless sensor networks," *IEEE Commun. Surveys & Tutorials*, vol. 12, no. 2, pp. 222–248, 2010.

[10] P. Suriyachai, U. Roedig, and A. Scott, "A survey of MAC protocols for mission-critical applications in wireless sensor networks," *IEEE Commun. Surveys & Tutorials*, vol. 14, no. 2, pp. 240–264, 2012.

[11] M. Bagaa, M. Younis, A. Ouadjaout, and N. Badache, "Efficient multi-path data aggregation scheduling in wireless sensor networks," in *Proc. IEEE ICC'13*, June 2013.

[12] Y. Bo, L. J, and Y. Li, "Distributed data aggregation scheduling in wireless sensor networks," in *Proc. IEEE INFOCOM'09*, April 2009, pp. 2159–2167.

[13] A. A. Abbasi and M. Younis, "A survey on clustering algorithms for wireless sensor networks," *Elsevier Computer Communications*, vol. 30, no. 14-15, pp. 2826 – 2841, 2007.

[14] J. Blum, M. Ding, A. Thaeler, and X. Cheng, *Handbook of Combinatorial Optimization*. Springer US, 2005, ch. Connected Dominating Set in Sensor Networks and MANETs, pp. 329–369.

[15] D. Eppstein, *Handbook of Computational Geometry*. Elsevier, 2000, ch. Spanning trees and spanners, pp. 425–461.

[16] M. Rahimi, R. Baer, O. I. Iroezi, J. C. Garcia, J. Warrior, D. Estrin, and M. Srivastava, "Cyclops: In situ image sensing and interpretation in wireless sensor networks," in *Proc. ACM SenSys'05*, November 2005, pp. 192–204.

[17] Memsic. [Online]. Available: http://www.memsic.com/wireless-sensor-networks

[18] Cc2420 single-chip 2.4 ghz ieee 802.15.4 compliant and zigbee(tm) ready rf transceiver. [Online]. Available: http://www.ti.com/lit/gpn/cc2420

[19] M. Doudou, D. Djenouri, and N. Badache, "Survey on latency issues of asynchronous mac protocols in delay-sensitive wireless sensor networks," *IEEE Commun. Surveys & Tutorials*, vol. 15, no. 2, pp. 528–550, 2013.

[20] I. Solis and K. Obraczka, "The impact of timing in data aggregation for sensor networks," in *Proc. IEEE ICC'04*, June 2004, pp. 3640–3645.

[21] H. Li, H. Yu, B. Yang, and A. Liu, "Timing control for delay-constrained data aggregation in wireless sensor networks: Research articles," *John Wiley and Sons Ltd. Int. J. Commun. Syst.*, vol. 20, pp. 875–887, 2007.

[22] X. Chen, X. Hu, and J. Zhu, "Minimum data aggregation time problem in wireless sensor networks," in *Proc. Springer-Verlag MSN'05*, 2005, pp. 133–142.

[23] S. Huang, P. Wan, C. Vu, Y. Li, and F. Yao, "Nearly constant approximation for data aggregation scheduling in wireless sensor networks," in *Proc. IEEE INFOCOM'07*, May 2007, pp. 366–372.

[24] B. Malhotra, I. Nikolaidis, and M. Nascimento, "Aggregation convergecast scheduling in wireless sensor networks," *Springer Wireless Networks*, vol. 17, no. 2, pp. 319–335, 2010.

[25] P. Wang, Y. He, and L. Huang, *Wireless Algorithms, Systems, and Applications*, ser. Lecture Notes in Computer Science, 2010, vol. 6221, ch. Approaching the Optimal Schedule for Data Aggregation in Wireless Sensor Networks, pp. 26–35.

[26] J. Zhu and X. Hu, "Improved algorithm for minimum data aggregation time problem in wireless sensor networks," *Springer Journal of Systems Science and Complexity*, vol. 21, no. 4, pp. 626–636, 2008.

[27] X. Xu, X.-Y. Li, X. Mao, S. Tang, and S. Wang, "A delay-efficient algorithm for data aggregation in multihop wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 1, pp. 163–175, 2011.

[28] P.-J. Wan, S. C.-H. Huang, L. Wang, Z. Wan, and X. Jia, "Minimum-latency aggregation scheduling in multihop wireless networks," in *Proc. ACM MobiHoc'09*, May 2009, pp. 185–194.

[29] M. Bagaa, A. Derhab, N. Lasla, A. Ouadjaout, and N. Badache, "Semi-structured and unstructured data aggregation scheduling in wireless sensor networks," in *Proc. IEEE INFOCOM'12*, March 2012, pp. 2671–2675.

[30] O. Incel and B. Krishnamachari, "Enhancing the data collection rate of tree-based aggregation in wireless sensor networks," in *Proc. IEEE SECON'08*, June 2009, pp. 569–577.

[31] S. Upadhyayula, V. Annamalai, and S. K. Gupta, "Enhancing the data collection rate of tree-based aggregation in wireless sensor networks," in *Proc. IEEE GLOBECOM'03*, December 2003, pp. 3525–3530.

[32] S. Upadhyayula and S. K. S. Gupta, "Spanning tree based algorithms for low latency and energy efficient data aggregation enhanced convergecast (dac) in wireless sensor networks," *Elsevier Ad Hoc Network*, vol. 5, no. 5, pp. 626–648, 2007.

[33] Ö. D. Incel, A. Ghosh, B. Krishnamachari, and K. Chintalapudi, "Fast data collection in tree-based wireless sensor networks," *IEEE Trans. Mob. Comput.*, vol. 11, no. 1, pp. 86–99, 2012.

[34] A. Ghosh, Ö. D. Incel, B. Krishnamachari, and A. Vullikanti, "Multi-channel scheduling algorithms for fast aggregated convergecast in sensor networks," in *Proc. IEEE MASS'09*, October 2009, pp. 363–372.

[35] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Trans. Inf. Theory*, vol. 46, no. 2, pp. 388–404, 2000.

[36] J. Grönkvist and A. Hansson, "Comparison between graph-based and interference-based stdma scheduling," in *Proc. ACM MobiHoc'01*, October 2001, pp. 255–258.

[37] P. Shao-Liang, L. Shan-Shan, P. Yu-Xing, Z. Pei-Dong, and X. Nong, "A delay sensitive feedback control data aggregation approach in wireless sensor network," in *Proc. Springer-Verlag ICCS'07*, May 2007, pp. 393–400.

[38] C. Huifang, M. Hiroshi, O. Yoshitsugu, K. Tomohiro, and M. Tadanori, "Adaptive data aggregation for clustered wireless sensor networks," in *Proc. Springer-Verlag UIC'07*, July 2007, pp. 475–484.

[39] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *Proc. ACM MOBICOM*, August 2000, pp. 56–67.

[40] J. Y. Choi, J. Lee, K. Lee, S. Choi, W. H. Kwon, and H. S. Park, "Aggregation time control algorithm for time constrained data delivery

in wireless sensor networks," in *Proc. IEEE VTC'06*, May 2006, pp. 563–567.
[41] F. Hu and C. May, "Timing control during data aggregation operations of distributed sensor networks: Trade-off between query accuracy and response delay," *Information Technology Journal*, vol. 5, pp. 759–778, 2006.
[42] F. Hu, X. Cao, and C. May, "Optimized scheduling for data aggregation in wireless sensor networks," in *Proc. IEEE ITCC'05*, April 2005, pp. 557–561.
[43] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: a tiny aggregation service for ad-hoc sensor networks," *ACM SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 131–146, Winter 2002.
[44] W. Huangfu, Y. Liu, B. Duan, L. Sun, J. Ma, and C. Chen, "Eata: Effectiveness based aggregation time allocation algorithm for wireless sensor networks," in *Proc. IEEE ISCC'08*, July 2008, pp. 981–987.
[45] Y. Y. K. Shan Guo Quan, "Fast data aggregation algorithm for minimum delay in clustered ubiquitous sensor networks," in *Proc. IEEE ICHIT'08*, 2008, pp. 327–333.
[46] A. T. Parameswaran, M. I. Husain, and S. Upadhyaya, "Is rssi a reliable parameter in sensor localization algorithms: An experimental study," in *Proc. F2DA'09*, September 2009.
[47] A. Sivagami, K. Pavai, and D. Sridharan, "Latency optimized data aggregation timing model for wireless sensor networks," *International Journal of Computer Science Issues*, vol. 7, pp. 42–48, 2010.
[48] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proc. ACM SenSys'09*, November 2009, pp. 1–14.
[49] Z. Chen, S. Yang, X. Huang, and Y. Liu, "An adaptive feedback timing control algorithm of delay-constrained data aggregation in wireless sensor networks," in *Proc. IEEE CyberC'09*, October 2008, pp. 337–342.
[50] P. J. WAN, A. K, and O. FRIEDER, "Distributed construction of connected dominating set in wireless ad hoc networks," in *Proc. IEEE INFOCOM'02*, June 2002, pp. 1597–1604.
[51] N. Burri, P. von Rickenbach, and R. Wattenhofer, "Dozer: ultra-low power data gathering in sensor networks," in *Proc. ACM IPSN'07*, April 2007, pp. 450–459.
[52] B. Yu and J. Li, "Making aggregation scheduling usable in wireless sensor networks: An opportunistic approach," in *Proc. IEEE MASS'11*, October 2011, pp. 666–671.
[53] H. V. Luu and X. Tang, "An efficient multi-path data collection scheme in wireless sensor networks," in *Proc. IEEE ICDCSW'11*, June 2011, pp. 198–207.
[54] M. Bagaa, M. Younis, A. Ksentini, and N. Badache, "Multi-path multi-channel data aggregation scheduling in wireless sensor networks," in *Proc. IFIP Wireless Days'13*, November 2013.
[55] M. Bagaa, M. Younis, and N. Badache, "Efficient data aggregation scheduling in wireless sensor networks with multi-channel links," in *Proc. ACM MSWIM'13*, November 2013.

Miloud Bagaa is research assistant at the Research Center on Scientific and Technical Information (CERIST), Algeria. He is member of the Wireless Sensor Network team at DTISI Division. He is also a Ph.D student at the University of Science and Technology Houari Boumediene (USTHB). He got his engineering and post graduate diploma degrees respectively in 2005 and 2008 from the University of Science and Technology Houari Boumediene (USTHB). His research interests are data management and security in wireless sensor networks.

Yacine Challal is associate professor at Ecole nationale Supreieure dInformatique (Algeria). He leads the Network Security research team at the LMCS lab (Systems Design Methods Laboratory). He got his PhD and Master degrees respectively in 2005, and 2002 from Compigne University of Technology. His research interests include security in group communication, security in wireless mobile networks, wireless sensor networks, IoT, Cloud computing and fault tolerance in distributed systems.

Adlen Ksentini is an Associate Professor at the University of Rennes 1, France. He is a member of the INRIA Rennes team Dionysos. He received an M.Sc. in telecommunication and multimedia networking from the University of Versailles. He obtained his Ph.D. degree in computer science from the University of Cergy-Pontoise in 2005, with a dissertation on QoS provisioning in IEEE 802.11-based networks. His other interests include: future Internet networks, cellular networks, green networks, QoS, QoE and multimedia transmission. Dr. Ksentini is involved in several national and European projects (FP7 Alicante and FP6 Anemone) on QoS and QoE support in Future wireless and mobile Networks. Recently, Dr. Ksentini has launched a bilateral collaboration with Orange Labs. on Small Cell Networks. Dr. Ksentini is a co-author of over 60 technical journal and international conference papers. He received Best Paper Award from IEEE ICC 2012, and ACM MSWiM 2005. Dr. Ksentini has been in the technical program committee of major IEEE ComSoc conferences, ICC/Globecom, ICME, WCNC, PIMRC. Dr. Ksentini is a Senior IEEE member.

Abdelouahid Derhab received the Engineer, Master, and Ph.D. degrees in computer science from University of Sciences and Technology Houari Boummediene (USTHB), Algiers, in 2001, 2003, and 2007 respectively. He was a computer science engineer and a full-time researcher at CERIST (CEntre de Recherche sur lInformation Scientifique et Technique) in Algeria from 2002 to 2012. He is currently an assistant professor at the Center of Excellence in Information Assurance (COEIA), King Saud University. His interest research areas are: mobile and wireless networks, distributed algorithms, and network security.

Nadjib Badache was with CISTTT (Centre dInformation Scientifique et Technique et de Transfert Technologique), now CERIST (Centre de Recherche sur lInformation Scientifique et Technique) (Algiers), from 1978 to 1983, working on applied research projects in information retrieval, operating systems and library management. In 1983, he joined USTHB (University of Algiers) as assistant professor and then professor, where he taught operating systems design, distributed systems and networking with research mainly in distributed algorithms and mobile systems. From 2000 to 2008, he was head of LSI (Laboratoire des Systmes Informatiques of USTHB University) where he conducted many projects on routing protocols, energy efficiency and security in mobile ad-hoc networks and wireless sensor networks. Since March, 2008 he is director of CERIST and professor at USTHB University.