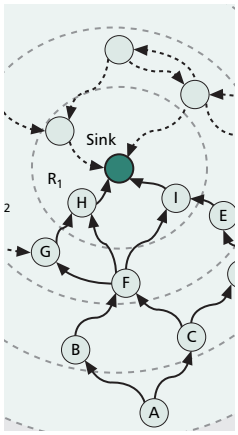
ACCEPTED FROM OPEN CALL

IN-NETWORK AGGREGATION TECHNIQUES FOR WIRELESS SENSOR NETWORKS: A SURVEY

ELENA FASOLO AND MICHELE ROSSI, DEI, UNIVERSITY OF PADOVA
JÖRG WIDMER, DOCOMO EURO-LABS
MICHELE ZORZI, DEI, UNIVERSITY OF PADOVA



The authors provide a comprehensive review of the existing literature on techniques and protocols for in-network aggregation in wireless sensor networks.

ABSTRACT

In this article we provide a comprehensive review of the existing literature on techniques and protocols for in-network aggregation in wireless sensor networks. We first define suitable criteria to classify existing solutions, and then describe them by separately addressing the different layers of the protocol stack while highlighting the role of a cross-layer design approach, which is likely to be needed for optimal performance. Throughout the article we identify and discuss open issues, and propose directions for future research in the area.

INTRODUCTION

Recent advances in technology make it feasible to mass produce small sensor nodes with sensing, computation, and communication capabilities. This has spurred a substantial amount of research on wireless sensor networks over the past few years. For ease of deployment, sensor devices should be inexpensive, small, and have a long lifetime, which makes it important to develop very efficient software and hardware solutions. For this reason, protocols for sensor networks should be carefully designed so as to make the most efficient use of the limited resources in terms of energy, computation, and storage. These restrictions are likely to remain, since in many cases it is desirable to exploit technological improvements to develop smaller and more energy efficient devices rather than making them more powerful. Typical applications envisioned for sensor networks (e.g., environmental monitoring, surveillance, tracking), along with the already mentioned resource-constrained character of sensor nodes, usually result in very different network requirements and communications patterns from other types of ad hoc network scenarios. The area of communications and protocol design for sensor networks has been widely researched in the past few years, and many solutions have been proposed and compared.

In this survey article we focus instead on another important aspect of sensor networks: in-network aggregation and data management. These techniques allow trading off communication for computational complexity. Given the application area, network resource constraints, and the fact that local computation often consumes significantly less energy than communication, in-network data aggregation and management are at the very heart of sensor network research. In particular, resource efficiency, timely delivery of data to the sink node, and accuracy or granularity of the results are conflicting goals, and the optimal trade-off among them largely depends on the specific application.

Initially, in-network aggregation techniques involved different ways to route packets in order to combine data coming from different sources but directed toward the same destination(s). In other words, these protocols were simply routing algorithms that differed from more traditional ad hoc routing protocols in the metric they used to select the routing paths. More recently, many additional studies have been published, addressing not only the routing problem but also mechanisms to represent and combine data more efficiently. In-network data aggregation is a complex problem that involves many layers of the protocol stack and different aspects of protocol design, and a characterization and classification of concepts and algorithms is still lacking in the literature.

The aim of the present article is to provide a taxonomy of in-network aggregation by defining the main concepts, and covering the most important and recent work in the field. Our major contributions are, on one hand, to define criteria to classify existing solutions and, on the other hand, to identify and propose directions for future research in this area. Compared to well researched topics in sensor networks, such as medium access control (MAC) and routing protocol design, data aggregation does not seem to have received as much attention, and we think it provides many interesting opportunities for relevant contributions. The goal of this article is to help people get an updated view of this area, and provide a motivation and a starting point for researchers and students who are interested in these issues.

The article is organized as follows. We define the in-network aggregation paradigm, by identifying the main problems involved and giving some

1536-1284/07/$20.00 © 2007 IEEE

criteria to classify existing algorithms. We discuss theoretical performance limits of in-network aggregation techniques. We introduce some protocol issues in the presence of in-network processing, classify the most recent solutions, and discuss their advantages and disadvantages. We focus on possible techniques to combine data by means of aggregation functions, highlight how these interact with routing protocols, and discuss the benefits arising from a cross-layer design (routing and aggregation). Finally, we summarize the in-network aggregation approaches discussed throughout the article, and give directions and motivations for future research.

Basics of In-Network Aggregation

In typical sensor network scenarios, data is collected by sensor nodes throughout some area, and needs to be made available at some central sink node(s), where it is processed, analyzed, and used by the application. In many cases, data generated by different sensors can be jointly processed while being forwarded toward the sink (e.g., by fusing together sensor readings related to the same event or physical quantity, or locally processing raw data before it is transmitted). In-network aggregation deals with this distributed processing of data within the network. Data aggregation techniques are tightly coupled with how data is gathered at the sensor nodes as well as how packets are routed through the network, and have a significant impact on energy consumption and overall network efficiency (e.g., by reducing the number of transmissions or the length of the packets to be transmitted). In-network data aggregation can be considered a relatively complex functionality, since the aggregation algorithms should be distributed in the network and therefore require coordination among nodes to achieve better performance. Also, we emphasize that data size reduction through in-network processing shall not hide statistical information about the monitored event. For instance, when multiple sensors collaborate in observing the same event, the number of nodes reporting it and the timing of the reports may reveal the event's size and/or dynamics, respectively.

We define the in-network aggregation process as follows:

In-network aggregation is the global process of gathering and routing information through a multihop network, processing data at intermediate nodes with the objective of reducing resource consumption (in particular energy), thereby increasing network lifetime.

We can distinguish between two approaches:

- *In-network aggregation with size reduction* refers to the process of combining and compressing data coming from different sources in order to reduce the information to be sent over the network. As an example, assume that a node receives two packets from two different sources containing the locally measured temperatures. Instead of forwarding the two packets, the sensor may compute the average of the two readings and send it in a single packet.
- *In-network aggregation without size reduction* refers to the process of merging packets coming from different sources into the same packet without data processing: assume receiving two packets carrying different physical quantities (e.g., temperature and humidity). These two values cannot be processed together, but they can still be transmitted in a single packet, thereby reducing overhead.

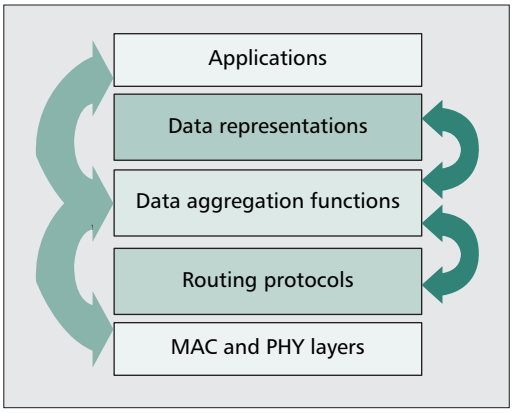


Figure 1. *Diagram for in-network aggregation techniques and their relation with different protocol layers. We stress that in general data processing also interacts with the application, MAC, and PHY layers.*

The first approach is better able to reduce the amount of data to be sent over the network, but it may also reduce the accuracy with which the gathered information can be recovered at the sink. After the aggregation operation, it is usually not possible to perfectly reconstruct all of the original data. This actually depends on the type of aggregation function in use (i.e., lossy or lossless.) The second approach instead preserves the original information (i.e., at the sink, the original data can be perfectly reconstructed). Which solution to use depends on many factors including the type of application, data rate, network characteristics, and so on. Both of the above strategies may involve the treatment of data at different network layers.

In-network aggregation techniques require three basic ingredients: suitable *networking protocols*, effective *aggregation functions*, and efficient ways of *representing the data* (Fig. 1). In the remainder of this section we briefly introduce each of these aspects.

Routing Protocols — The most important ingredient for in-network aggregation is a well designed routing protocol [1–9]. Data aggregation requires a different forwarding paradigm than classic routing. Classic routing protocols typically forward data along the shortest path to the destination (with respect to some specified metric). If, however, we are interested in aggregating data to minimize energy expenditure, nodes should route packets based on packet content and choose the next hop in order to promote in-network aggregation. This type of data forwarding is often referred to as *data-centric routing*. According to the data-centric paradigm, as a node searches for relay nodes, it needs to use metrics that take into account the positions of the most suitable aggregation points, the data type, the priority of the information, and so on.

Data aggregation techniques are tightly coupled with how data is gathered at the sensor nodes as well as how packets are routed through the network, and have a significant impact on overall network efficiency (e.g., by reducing the number of transmissions or the length of the packets to be transmitted).

Good aggregation functions for wireless sensor networks need to meet additional requirements. In particular, they should take into account the very limited processing and energy capabilities of sensor devices, and should therefore be implementable by means of elementary operations.

Altogether, the application scenario, routing scheme, and data aggregation mechanism are closely interrelated.

Moreover, in-network aggregation techniques may require some form of synchronization among nodes. In particular, the best strategy at a given node is not always to send data as soon as it is available. Waiting for information from neighboring nodes may lead to better data aggregation opportunities and, in turn, improved performance. Timing strategies are required especially in the case of monitoring applications where sensor nodes need to periodically report their readings to the sink. These strategies usually involve data gathering trees rooted at the sink. The main timing strategies proposed so far in the literature are summarized below [10]:

- *Periodic simple aggregation* requires each node to wait for a predefined period of time, aggregate all data items received, and then send out a packet with the result of the aggregation.
- *Periodic per-hop aggregation* is quite similar to the previous approach, the only difference being that the aggregated data is transmitted as soon as the node hears from all of its children. This requires that each node knows the number of its children. In addition, a timeout is used in case some children's packets are lost.
- *Periodic per-hop adjusted aggregation* adjusts the timeout of a node, after which it sends the aggregated data, depending on the node's position in the gathering tree. Note that the choice of the timing strategy strongly affects the design of the routing protocol [10–12].

Aggregation Functions — One of the most important functionalities that in-network aggregation techniques should provide is the ability to combine data coming from different nodes. There are several types of aggregation functions [8, 13–20], and most of them are closely related to the specific sensor application. Nevertheless, we can identify some common paradigms for their classification:

- *Lossy and lossless*: Aggregation functions can compress and merge data according to either a lossy or a lossless approach. In the first case the original values cannot be recovered after having merged them by means of the aggregation function. In addition, we may lose precision with respect to transmitting all readings uncompressed. In contrast, the second approach (lossless) allows us to compress the data by preserving the original information. This means that all readings can be perfectly reconstructed from their aggregate at the receiver side.
- *Duplicate sensitive and duplicate insensitive*: An intermediate node may receive multiple copies of the same information. In this case, it may happen that the same data is considered multiple times when the information is aggregated. If the aggregation function in use is duplicate sensitive, the final result depends on the number of times the same value has been considered. Otherwise, the aggregation function is said to be duplicate insensitive. For instance, a function that takes the average is duplicate sensitive, whereas a function that takes the minimum value is duplicate insensitive.

Good aggregation functions for wireless sensor networks need to meet additional requirements. In particular, they should take into account the very limited processing and energy capabilities of sensor devices, and should therefore be implementable by means of elementary operations. Also, different devices may be suitable for different types of operations, depending on their energy resources and computation capabilities. These facts need to be considered in the design of aggregation functions and routing protocols.

Data Representation — Due to its limited storage capabilities, a node may not be able to store all the received/generated information in its internal buffer. It therefore needs to decide whether to store, discard, compress, or transmit the data. All these operations require a suitable way to represent the information [21–24]. The corresponding data structure may vary according to the application requirements. Finally, even though the data structure is usually common to all nodes, it should be adaptable to node-specific or location specific characteristics. A recent and promising method to deal with data representation and compression is distributed source coding techniques that compress data on the basis of some knowledge about its correlation. More details on the approach are given later.

Although we described routing, aggregation, and data representation in isolation, they are intimately related and should be designed and implemented jointly for optimal performance. Most of the related work in the literature covers only partial aspects of the joint optimization of these functionalities, and often neglects or oversimplifies some of the others. Further work on cross-layer optimization for in-network aggregation should therefore be appreciated as innovative and is very much needed. In the rest of the article we thoroughly review each of the aforementioned functionalities. We start with a review of recent work on the theoretical limits of aggregation techniques in the next section.

Theoretical Limits of In-Network Aggregation Techniques

Several theoretical studies provide limits and bounds on the performance of in-network data aggregation techniques and thus assist in the design of suitable algorithms. The efficiency of these algorithms depends on the correlation among the data generated by different information sources (sensor units). Such a correlation can be spatial, when the values generated by close-by sensors are related; temporal, when the sensor readings change slowly over time; or semantic, when the contents of different data packets can be classified under the same semantic group (e.g., the data is generated by sensors placed in the same room). The gains of in-network data aggregation can best be demonstrated in the extreme case when data generated by different sources can be combined into a single packet (e.g., when the sources generate identical data). If there are K sources all close to each other and far away from the sink, the combination of their data into a single packet leads, on average, to a K-fold reduc-

tion in transmissions with respect to the case where all data are sent separately. Generally, the optimal joint routing and compression structure is a Steiner tree, which is known to be NP-hard [25]. However, there exist polynomial solutions for special cases where the information sources are close to each other [26]. The authors in [27] propose a model to describe the spatial correlation in terms of joint entropy. They analyze a symmetric line network with different degrees of correlation among neighboring nodes. For the uncorrelated case, the authors show that the best routing strategy is to forward packets along shortest paths. In contrast, in case of completely correlated information, the best strategy is to aggregate data as soon as possible. After that, a single packet (formed by the aggregated data) is sent to the sink along the shortest path. In all the intermediate cases, clustering-based solutions may be the optimal choice, although no formal proof is given in the article.

In [28] the authors study the impact of data correlation on the energy expenditure of data distribution protocols. They focus on various energy aware data aggregation trees under different network conditions, such as node density, source density, source distribution, and data aggregation degree. The findings of the study are in agreement with the results in [27] but, in addition, provide more quantitative results. In particular, the authors focus on tree structures and compare the minimum Steiner tree (MST) with the shortest path tree (SPT). The MST is found to be the optimal aggregation tree structure. Although the SPT guarantees low delays and can be built in an online fashion, its performance in terms of aggregation effectiveness is largely inferior to that of the MST.

In addition, in [28] opportunistic aggregation is compared to systematic aggregation in terms of cost ratio, which is the cost of the *correlation-unaware* (SPT) tree over that of the *correlation-aware* (MST) tree considering the same set of sources and sinks. The authors prove, using an analytical model, that the expected cost improvement of MST over SPT in sensor networks increases as

$$O\left(\sqrt{\log N}\right),$$

where N is the number of nodes in the network. This result makes SPT a viable solution for many practical cases (small networks). Based on this study, the authors propose a hybrid tree structure called *semantic/spatial correlation tree* (SCT) [29]. SCT is based on the identification of an aggregation backbone which is used to generate efficient aggregation trees, regardless of sources distribution and density. The aim is to efficiently build and maintain a network structure for data aggregation. To this end, the authors of [29] propose a ring-sector subdivision of the network. A subset of nodes is elected as aggregation nodes, and they are organized in a spanning tree to form the data aggregation backbone. Each node belonging to the backbone aggregates messages coming from a certain subarea.

A further tree-based aggregation algorithm that exploits data correlation is presented in [30]. It is based on shallow light trees (SLTs) that unify the properties of MSTs and SPTs. In an SLT, the total cost of the tree is only a constant factor larger than that of the MST, while the distances (delays) between any node and the sink are only a constant factor larger than the shortest paths. In [31] the authors analyze aggregation properties of a tree structure that is based on an SPT of nodes close to the sink node, while nodes that are further away are connected to the leaves of the SPT via paths found by an approximation algorithm for the traveling salesman problem. Simulations show that these trees outperform SLTs in many scenarios.

Networking Protocols and Hierarchies for In-Network Aggregation

Most of the work done so far on in-network aggregation deals with the problem of forwarding packets in order to facilitate the in-network aggregation of the information therein. Initially, the main ideas were to enhance existing routing algorithms in such a way as to make data aggregation possible. To this end, many studies proposed solutions exploiting *tree-based* (or *hierarchical*) structures. These consist of routing algorithms based on a tree rooted at the sink. Trees are usually SPTs, but some approaches consider more complex tree constructions. The tree-based approaches are referred to in this article as *classical approaches*. Sometimes the tree structure can be optimized to the type of data to be gathered. Also, the nodes can be locally grouped into clusters for improved efficiency. Recently, a few notable exceptions looked at the problem from a different angle. These papers address the weaknesses of the tree-based approach by focusing on *multipath* routing. Finally, some very recent schemes implement a mixture of tree-based and multipath solutions. These are referred to here as *hybrid approaches* to emphasize the adaptive nature of their routing algorithms.

In the following, we focus on each class of routing protocols separately (tree-based, cluster-based, multipath, and hybrid) by reviewing the main concepts and briefly commenting on the pros and cons of each scheme. As seen from the number of schemes discussed in each subsection, many solutions are proposed in the tree-based and cluster-based categories. On the other hand, very few studies use the multipath and hybrid approaches. This leaves room for further work in this area.

Tree-Based Approaches

Classic routing strategies [32, 33] are usually based on a hierarchical organization of the nodes in the network. In fact, the simplest way to aggregate data flowing from the sources to the sink is to elect some special nodes that work as aggregation points and define a preferred direction to be followed when forwarding data.

In addition, a node may be marked as special depending on many factors such as its position within the data gathering tree [34], its resources [35], the type of data stored in its queue [36, 37], or the processing cost due to aggregation procedures [38]. According to the *tree-based approach* [1, 3, 6] a spanning tree rooted at the sink is

The simplest way to aggregate data flowing from the sources to the sink is to elect some special nodes that work as aggregation points and define a preferred direction to be followed when forwarding data.

The TAG approach is a data centric protocol. It is based on aggregation trees and is specifically designed for monitoring applications. This means that all nodes should produce relevant information periodically. Therefore, it is possible to classify TAG as a periodic per hop adjusted aggregation approach.

constructed first. Subsequently, such a structure is exploited in answering queries generated by the sink. This is done by performing in-network aggregation along the *aggregation tree* by proceeding level by level from its leaves to its root. Thus, as two or more messages get to a given node, their aggregate can be computed exactly. However, this way of operating has some drawbacks as actual wireless sensor networks are not free from failures. More precisely, when a packet is lost at a given level of the tree (e.g., due to channel impairments), the data coming from the related subtree are lost as well. In fact, a single message at a given level of the tree may aggregate the data coming from the whole related subtree. In spite of the potentially high cost of maintaining a hierarchical structure in dynamic networks and the scarce robustness of the system in case of link/device failures, these approaches are particularly suitable for designing optimal aggregation functions and performing efficient energy management. In fact, there are some studies where the sink organizes routing paths to evenly and optimally distribute the energy consumption while favoring the aggregation of data at the intermediate nodes [36, 39, 40]. In [39] the authors compute aggregation topologies by taking into account the residual energy of each node through linear programming. Further algorithms can be found in [34, 35, 41, 42]. In [41] the authors investigate which nodes in the network can be exploited as aggregation points for optimal performance. In [34, 42] the focus is on the nodes that should be entrusted with the transmission of the sensed values, whereas in [35] the emphasis is put on the proper scheduling of sleeping/active periods. Often, optimal paths are calculated in a centralized manner at the sink by exploiting different assumptions on the data correlation and selecting the best aggregation points by means of cost functions [43]. Recently, tree-based schemes for real-time or time-constrained applications have also been proposed [44–46].

Finally, a last approach based on aggregation trees relies on the construction of *connected dominating sets* [47]. These sets consist of a small subset of nodes that form a connected backbone and whose positions are such that they can collect data from any point in the network. Nodes that do not belong to these sets are allowed to sleep when they do not have data to send. Some rotation of the nodes in the dominating set is recommended for energy balancing.

In the following paragraphs we review the main routing approaches based on aggregation trees.

TAG — The *Tiny AGgregation* (TAG) approach [5] is a data-centric protocol. It is based on aggregation trees and specifically designed for *monitoring applications*. This means that all nodes should produce relevant information periodically. Therefore, it is possible to classify TAG as a *periodic per hop adjusted* aggregation approach. The implementation of the core TAG algorithm consists of two main phases:

- The *distribution phase*, where queries are disseminated to the sensors
- The *collection phase*, where the aggregated sensor readings are routed up the aggregation tree

For the *distribution phase*, TAG uses a tree-based routing scheme rooted at the sink node. The sink broadcasts a message asking nodes to organize into a routing tree and then sends its queries. In each message there is a field specifying the level, or distance from the root, of the sending node (the level of the root is equal to zero). Whenever a node receives a message and it does not yet belong to any level, it sets its own level to be the level of the message plus one. It also elects the node from which it receives the message as its parent. The parent is the node that is used to route messages toward the sink. Each sensor then rebroadcasts the received message adding its own identifier (ID) and level. This process continues until all nodes have been assigned an ID and a parent. The routing messages are periodically broadcast by the sink in order to keep the tree structure updated. After the construction of the tree, the queries are sent along the structure to all nodes in the network. TAG adopts the selection and aggregation facilities of the database query languages (SQL). Accordingly, TAG queries have the following form:

```
SELECT{agg(expr), attrs} from SENSOR
   WHERE{selPreds}
   GROUP BY{attrs}
   HAVING{havingPreds}
   EPOCH DURATION i
```

In practice, the sink sends a query, where it specifies the quantities that it wants to collect (`attrs` field), how these must be aggregated (`agg(expr)`), and the sensors that should be involved in the data retrieval. This last request is specified through the `WHERE`, `GROUP`, and `HAVING` clauses [5]. Finally, an `EPOCH DURATION` field specifies the time (in seconds) each device should wait before sending new sensor readings. This means the readings used to compute an aggregate record all belong to the same time interval, or epoch.

During the data *collection phase*, due to the tree structure, each parent has to wait for data from all of its children before it can send its aggregate up the tree. Epochs are divided into shorter intervals called *communication slots*. The number of these slots equals the maximum depth of the routing tree. The slot mechanism gives a nice benefit. As the time is slotted, sensor nodes can be put to sleep until the next scheduled transmission interval. In practice, a node goes back to sleep soon after it has finished sending its readings to its parent. Data aggregation is performed by all intermediate nodes. However, in order not to limit TAG to the few and very simple aggregation functions defined by the SQL language (e.g., COUNT, MIN, MAX, SUM, and AVERAGE) a more general classification is accounted for by partitioning aggregates according to the *Duplicate Sensitivity*, *Exemplary and Summary*, and *Monotonic* properties [5].

As for most tree-based schemes, TAG may be inefficient for dynamic topologies or link/device failures: as discussed above, trees are particularly sensitive to failures at intermediate

nodes as the related subtree may become disconnected. In addition, as the topology changes, TAG has to reorganize the tree structure, which means high costs in terms of energy consumption and overhead.

Directed Diffusion — *Directed Diffusion* [1] is a reactive data-centric protocol. The routing scheme is specifically tailored for those applications where one or few sinks ask some specific information by flooding the network with their queries. Directed Diffusion is organized in three phases (see Fig. 2, originally shown in [1]):

- *Interest dissemination*
- *Gradient setup*
- Data forwarding along the reinforced paths (*path reinforcement* and *forwarding*)

When a certain sink is interested in collecting data from the nodes in the network, it propagates an *interest* message (*interest dissemination*), describing the type of data in which the node is interested, and setting a suitable operational mode for its collection. Each node, on receiving the interest, rebroadcasts it to its neighbors. In addition, the node sets up *interest gradients*, that is, vectors containing the next hop that has to be used to propagate the result of the query back to the sink node (*gradient setup*). As an illustrative example (Fig. 2), if the sink sends an interest that reaches nodes *a* and *b*, and both forward the interest to node *c*, node *c* sets up two vectors indicating that the data matching that interest should be sent back to *a* and/or *b*. The strength of such a gradient can be adapted, which may result in a different amount of information being redirected to each neighbor. To this end, various metrics such as the node's energy level, communication capability, and position within the network can be used. Each gradient is related to the attribute for which it has been set up. As the gradient setup phase for a certain interest is complete, only a single path for each source is *reinforced* and used to route packets toward the sink (*path reinforcement* and *forwarding*).

Data aggregation is performed when data is forwarded to the sink by means of proper methods, which can be selected according to application requirements. The data gathering tree (i.e., reinforced paths) must be periodically refreshed by the sink, and this can be expensive in dynamic topologies. A trade-off, depending on the network dynamics, is involved between the frequency of the gradient setup (i.e., energy expenditure) and the achieved performance. A valuable feature of Directed Diffusion consists of the *local interaction* among nodes in setting up gradients and reinforcing paths. This allows for increased efficiency as there is no need to spread the complete network topology to all nodes in the network.

We observe that attention is to be paid to MAC layer design. Consider as an example the IEEE 802.11 wireless technology. As mentioned above, queries are propagated by means of broadcasts (basic access in IEEE 802.11). However, data is sent back to the sink via unicast transmissions. This means that when either the node density increases or the duplicate suppression rule is not used, due to MAC collisions and subsequent backoffs, the delay may become excessively large. Hence, the local traffic should be kept at an acceptably low level in order to avoid collisions. Several approaches [36, 48, 49] have been proposed to reduce the control traffic generated by the local interactions among nodes with Directed Diffusion. In these solutions the authors use properly defined aggregation trees with the main purpose of reducing both traffic and delay. In [48] a modified version of Directed Diffusion, *Enhanced Directed Diffusion* (EDD), is proposed. This protocol jointly exploits Directed Diffusion to collect data and a cluster-based architecture to increase the efficiency of the local interactions (decreasing local traffic and related collisions). A similar approach is investigated in [50].

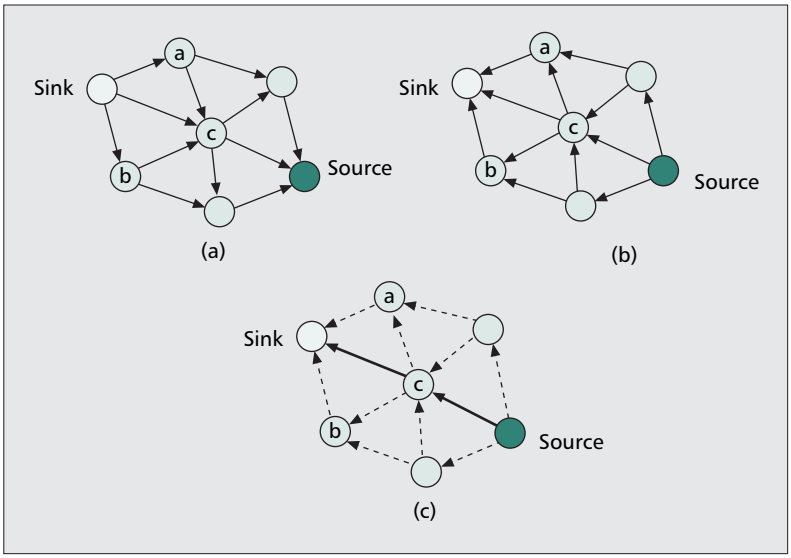


Figure 2. *A simplified scheme for Directed Diffusion [1]: a) interest dissemination; b) gradients setup; c) data delivery along the reinforced path.*

PEGASIS — The key idea in *Power-Efficient GAthering in Sensor Information Systems* (PEGASIS) [3] is to organize the sensor nodes in a chain. Moreover, nodes take turns acting as the chain leader, where at every instant the chain leader is the only node allowed to transmit data directly to the sink. In this way it is possible to evenly distribute the energy expenditure among the nodes in the network. The chain can be built either in a centralized (by the sink) or distributed manner (by using a greedy algorithm at each node). In both cases, however, the construction of the chain requires global knowledge of the network at all nodes. The chain building process starts with the node furthest from the sink. Then the closest neighbor to this node is chosen as the next one in the chain, and so on. Nodes take turns acting as leader according to the following rule: node i is elected as the leader in round i. If there are N nodes in the network, rounds cyclically take values in $\{1, 2, \ldots, N\}$ according to a time-division multiple access (TDMA) schedule. As a consequence, the leader is not always the same, but during each transmission round it is at a different position in the chain. Note that in this scheme a direct communication channel from each sensor to the sink is required.

In PEGASIS, each node receives data from a neighbor and aggregates it with its own reading by generating a single packet of the same length.

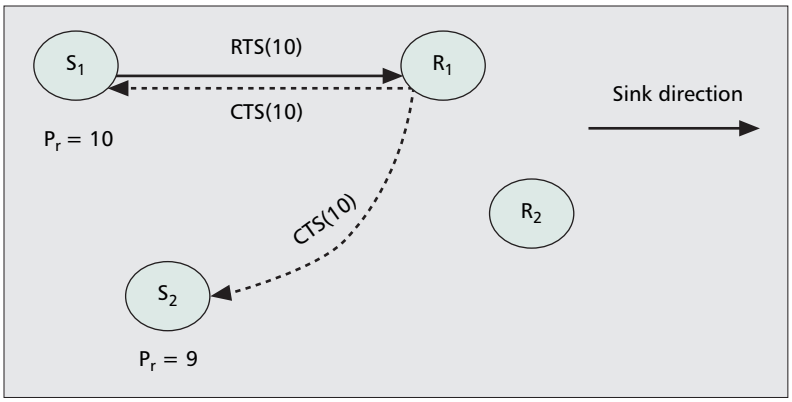


Figure 3. *A message exchange example in DB-MAC.*

Subsequently, such an aggregate is transmitted to the next node in the chain until the packet reaches the current chain leader. At this point the leader includes its own data into the packet and sends it to the sink. A possible drawback of the scheme comes from the distance among neighbors. In fact, when the neighbors along the chain are too distant, the energy expenditure can be very high. In addition, transmission energies are not evenly distributed but depend on the actual distances between the nodes and their neighbors (i.e., nodes with distant neighbors dissipate more energy). PEGASIS can therefore be enhanced by not allowing such nodes to become leaders, for example, using a threshold-based leader election policy. The main disadvantages of PEGASIS are the necessity of having a complete view of the network topology at each node for proper chain construction and that all nodes must be able to transmit directly to the sink. This makes the scheme unsuitable for those networks with a time varying topology. In addition, link failures and packet losses may also affect the performance of this protocol. In fact, the failure of any intermediate node compromises the delivery of all data aggregated and sent by the previous nodes in the chain. Hence, some improvements to the scheme may be needed in order to increase its robustness.

DB-MAC — A different approach, routing packets by performing data aggregation, is presented in [7], where the routing and MAC protocols are jointly designed. The primary objective of the *Delay Bounded Medium Access Control* (DB-MAC) [7] scheme is to minimize the latency for delay bounded applications while taking advantage of data aggregation mechanisms for increased energy efficiency. DB-MAC adopts a carrier sense multiple access with collision avoidance (CSMA/CA) contention scheme based on a request to send/clear to send/data/acknowledgment (RTS/CTS/DATA/ACK) handshake. The protocol is most suitable for those cases where different sources sense an event almost at the same time and, due to delay constraints, have to send their measurements right away to the sink. In such cases the generated data flows can be dynamically aggregated while routing them toward the sink. This gives rise to an aggregation tree, which is built on the fly and with no knowledge of the network topology. The MAC protocol is very similar to the IEEE 802.11 RTS/CTS Access [51] with some minor modifications: RTS/CTS messages are exploited to perform data aggregation, and backoff intervals are computed by taking into account the priorities assigned to different transmissions. In particular, each node takes advantage of the transmissions from other nodes by overhearing CTSs in order to facilitate data aggregation. This leads to choosing the relay node among those nodes that already have some packets to transmit in their queue. This is implemented to promote data aggregation with the information stored along the path.

As an example, refer to the scenario in Fig. 3. We have two nodes, S_1 and S_2, which want to transmit their packets to the sink using one of their neighbors (R_1 and R_2 in the figure) as the relay. At the beginning of the contention, a node transmits a newly generated packet by setting its priority to the maximum value. The packet priority is subsequently decreased at each traversed node. Because $P_r(S_1) > P_r(S_2)$, S_1 wins the contention for the medium and sends its packet to R_1, which decreases its priority by one unit. After this, $P_r(S_2)$ becomes equal to the priority of the packet just transmitted, which is now stored at node R_1. If S_2 is placed in the coverage area of both S_1 and R_1, it can overhear all messages exchanged between these two nodes (remember that the packet at S_2 still has to be forwarded). If this is the case, S_2 may now want to send its packet to R_1 instead of R_2 as it knows that R_1 already has one packet in its queue (the packet previously transmitted by S_1). This facilitates in-network aggregation. DB-MAC gives an example of how routing and data aggregation may influence each other, and shows that, in most cases, energy-efficient solutions are achieved only through a cross-layer design. The advantage of this strategy is the flexible and distributed procedure for the construction of aggregation trees, which appears to be suitable for wireless networks with dynamic topology.

Further Algorithms — Regarding the tree-based approaches, many additional solutions have been proposed to solve the problem of efficiently constructing aggregation trees. The authors in [36] define efficient, distributed, and energy-aware heuristics (EADAT) to build the aggregation tree. A nice feature of such an approach is that the tree construction process only relies on local knowledge of the network topology. Hence, the costs incurred in updating the tree in response to node mobility, device failures, and duty cycles may be limited. In addition, to further increase the energy savings, the scheme in [36] uses an aggregation tree rooted at the sink where all non-leaf sensors perform data aggregation while leaf nodes can turn off their radios in order to save energy. In [52] the problem of constructing the optimal aggregation tree is treated from a game theoretic perspective. The authors develop a framework including payoff functions that take into account path reliability, path length, and the energy constraints of the nodes. They finally propose and evaluate a couple of heuristics to implement opportunistic in-network aggregation strategies. Reference [53] presents a solution for the mobile sink case. The authors define a pro-

tocol to maintain the aggregation tree in the presence of mobile sinks. In their solution they rely on trusted nodes that work as gateways between the network and the sinks. A further contribution can be found in [54], where the authors combine a tree-based scheme with data compression based on polynomial regression.

An additional problem related to the aggregation tree is addressed in [37], where the authors present a set of algorithms to minimize the overall energy consumption of the sensor nodes in the presence of latency constraints under the assumption of perfect knowledge of the aggregation tree. The problem is solved by devising appropriate scheduling strategies for each node. This contribution is particularly important for applications requiring prompt delivery of the information to the sink. A major drawback, however, is that the problem of constructing the aggregation tree is not addressed. Finally, in [55] the *Secure Data Aggregation Protocol* (SDAP) scheme is presented. This algorithm addresses the problem of delivering data over aggregation trees in a secure manner. Further data aggregation schemes for secure communications are reviewed in [56].

Cluster-Based Approaches

Similar to tree-based algorithms, *cluster-based schemes* [2, 4, 48, 57] also consist of hierarchical organization of the network. However, here nodes are subdivided into clusters. Moreover, special nodes, referred to as *cluster heads*, are elected in order to aggregate data locally and transmit the result of such aggregation to the sink. The advantages and disadvantages of cluster-based schemes are very similar to those of tree-based approaches.

LEACH — *Low-Energy Adaptive Clustering Hierarchy* (LEACH) [2] is a self-organizing and adaptive clustering protocol using randomization to evenly distribute the energy expenditure among the sensors. Clustered structures are exploited to perform data aggregation where cluster heads act as aggregation points. The protocol works in rounds and defines two main phases:

- A *setup phase* to organize the clusters
- A *steady-state phase* that deals with the actual data transfers to the sink node

In the first phase the nodes organize themselves into clusters. Within each cluster a node is elected as the cluster head. At the beginning of the setup phase, each sensor elects itself to be the local cluster head for the current round. This decision is made according to a distributed probabilistic approach. The aim is to have, on average, a percentage P of the nodes acting as cluster heads, where P has to be optimally chosen according to the node density. In practice, sensors calculate the following threshold:

$$T(n)=\begin{cases}\dfrac{P}{1-P(R \bmod (1/P))} & \text{if } n \in G \\ 0 & \text{otherwise,}\end{cases} \quad (1)$$

where P is the desired percentage of cluster heads, R is the round number, and G is the set of nodes that have not been cluster heads during the last $1/P$ rounds. A given node n picks a random number [0, 1] and decides to be a cluster head if this number is lower than $T(n)$. A cluster head sends advertisements to its neighbors using a CSMA MAC. Surrounding nodes decide which cluster to join based on the signal strength of these messages. Finally, based on the number of nodes that are willing to be part of the cluster, each cluster head creates a TDMA schedule to optimally manage the local transmissions.

The actual data transmission starts in the second phase of the protocol. All source nodes (S in Fig. 4) send their data to their cluster heads according to the established schedule. The use of a TDMA protocol in the data collection phase ensures that there are no collisions within the clusters, saving both energy and time. After cluster heads (CH in Fig. 4) have received all the data from the active sources, they send them back to the sink using a single direct transmission (dotted lines in Fig. 4). If the sink is placed far away from a cluster head, high power may be necessary to successfully deliver the message. Also, a doze mode is implemented to further save energy. When doze mode is used, the nodes' radios may be switched off until their scheduled TDMA transmission slot. Note that cluster heads cannot switch their radio off as they have to receive packets from potentially all nodes in the cluster. LEACH is completely distributed in the sense that neither control messages from the sink nor the distribution of global information to the sensor nodes are required for correct operation. Moreover, LEACH outperforms classical clustering algorithms by accounting for adaptive clusters and rotating cluster heads.

The LEACH framework also offers the opportunity to implement any aggregation function at the cluster heads. However, several problems may arise in highly dynamic environments. In this case continuous updates are needed in order to keep the clusters consistent with the underlying topology. This requires sending many control messages, which, in turn, may substantially impact performance. In addition, with mobility additional problems may arise. A node close to a cluster head at a given instant in time may move away from the cluster head. As a consequence, the node needs to increase its power, thereby spending much more energy to transmit to the cluster head than expected.

Cougar — *Cougar* [4, 58] is most suitable for monitoring applications, where nodes produce relevant information periodically. The protocol can be classified as a *periodic per hop* aggregation approach. Cougar is basically a clustering scheme. As soon as the cluster heads receive all data from the nodes in their clusters, they send their partial aggregates to a gateway node. Of course, being similar to LEACH, Cougar is also affected by the same problems in highly dynamic environments.

Noticeably, Cougar differs from the previous clustering-based algorithms in the way cluster heads are elected. Unlike in LEACH, where each node picks its cluster head based on signal strength measurements, in Cougar cluster head selection may be driven by additional metrics. In fact, a node could be more than one hop away from its cluster head. For this reason, the rout-

LEACH is completely distributed in the sense that neither control messages from the sink nor the distribution of global information to the sensor nodes are required for correct operation.

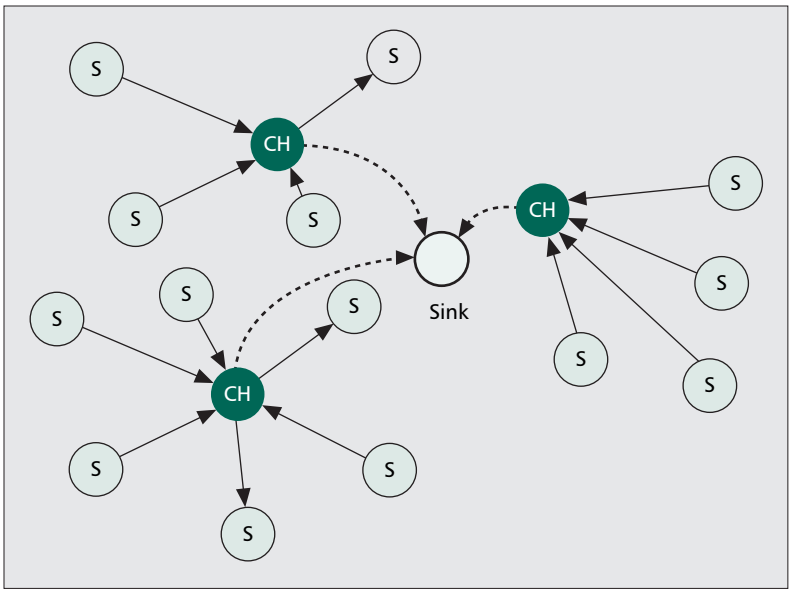


Figure 4. *LEACH clustering approach.*

ing algorithm adopted to exchange packets within clusters is based on the Ad Hoc On Demand Distance Vector (AODV) technique. As AODV does not generate duplicate data packets, Cougar is particularly suitable to perform in-network aggregation with duplicate sensitive aggregators. The core Cougar algorithm consists of the node synchronization engine, which ensures that data is aggregated correctly. Each cluster head has a waiting list containing all nodes from which it expects a message. The list is updated every time the node receives a record from a node in its cluster. The cluster head does not report its reading to the gateway until, at time t_{send}, it hears from all nodes in its waiting list. A prediction mechanism is also implemented at each cluster head in order to infer the instant t_{send}. In addition, a child node can determine whether its cluster head is waiting for a packet from it and can use a *notification packet* to refine the prediction at the cluster head. Timeouts and backoffs are implemented to deal with wrong predictions.

In [4] the authors define three different data aggregation features: *direct delivery*, where data aggregation is performed at the cluster-heads only, *packet merging*, which consists of aggregation of packets without size reduction, and *Partial aggregation*, where data aggregation is implemented at the intermediate nodes.

Further Algorithms — Many additional studies exploiting a hierarchical organization of the nodes have been proposed in the literature. Some of them are improvements of existing protocols. In [59], for instance, the authors propose enhancements to the LEACH and PEGASIS schemes. For performance evaluation, the authors propose the new data aggregation quality (DAQ) metric, which is defined as the ratio between the size of the aggregated data and its joint entropy. DAQ is an interesting performance measure as it takes into account both the effectiveness in reducing the size of the data to be transmitted and the quality of the information. A further improvement to LEACH is presented in [60], where a code is added to the data transmission to enhance intracluster communication security. A similar approach is proposed in [57] where the cluster-based scheme is enhanced by a secure transmission protocol called SecureDAV.

Reference [27] presents a location-based clustering scheme where the sensors self-organize to form static clusters. The data generated within each cluster is sent to the related cluster head along shortest paths, and in-network aggregation is performed at the intermediate nodes. The cluster heads send the aggregated data to the sink through a multihop path without any further aggregation. The cluster size can be varied to tune the degree of aggregation. The authors in [61] study the impact of partially correlated data on the performance of clustering algorithms. They analyze the behavior of multihop routing and, by combining random geometry techniques and rate distortion theory, predict the total energy consumption and network lifetime of their cluster-based scheme. Further cluster-based algorithms for data aggregation can be found in [62, 63]. Interesting work on clustering and data aggregation is presented in [64]. Here, a cross-layer approach is adopted, and some issues concerning MAC design are addressed.

Other work based on a hierarchical organization of the network is proposed in [11]. Assuming that some algorithms are used to form an aggregation tree or a cluster-based aggregation structure, the authors propose a scheme to dynamically adapt the data aggregation period (see [10]) according to the aggregation quality required by the sink.

A different approach is presented in [65] where a semi-structured approach named ToD is defined in order to alleviate the problem of maintaining a hierarchical organization of nodes in dynamic large-scale networks. This study is enriched by simulations with 2000 nodes and experimental results obtained from a large testbed.

Multipath Approaches

In order to overcome the robustness problems of aggregation trees, a new approach was recently proposed [8, 9, 66]. Instead of having an aggregation tree where each node has to send the partial result of its aggregation to a single parent, these solutions send data over multiple paths. The main idea is that each node can send the data to its (possibly) *multiple* neighbors by exploiting the broadcast characteristics of the wireless medium. Hence, data may flow from the sources to the sinks along multiple paths, and aggregation may be performed by each node. Observe that in contrast to the tree-based schemes discussed above, multipath approaches allow duplicates of the same information to be propagated. Clearly, such schemes trade higher robustness (as multiple copies of the same data can be sent along multiple paths) for some extra overhead (due to sending duplicates). An aggregation structure that fits well with this methodology is called *ring topology*, where sensor nodes are divided into several levels according to the number of hops separating them from the data sink. Data aggregation is performed over multiple paths as packets move level by level toward the sink (Fig. 5). Next, we review the synopsis

diffusion framework which belongs to this class of protocols.

Synopsis Diffusion — The authors of [8] present the *Synopsis Diffusion* protocol, where data aggregation is performed through a multipath approach. The underlying topology for data dissemination is organized in concentric rings around the sink. Synopsis Diffusion consists of two phases:

- The *distribution of the queries*
- The *data retrieval* phase

The ring topology is formed when a node sends a query over the network. In particular, two different structures can be taken into account. The first type of topology consists of a simple ring structure. During the query distribution phase, the network nodes form a set of rings around the querying node q, which is the only sensor belonging to ring R_0. A node is in ring R_i if it is i hops away from the querying node.

The second type of topology has some improvements that make it more robust than the first and able to cope with changes in the network. This topology is called *adaptive rings*. The distribution phase does not change, but this time a node u in ring i keeps track of the number of times, n_{ov}, the transmissions from any node n_{i-1} in ring $i - 1$ included its own data during the last few epochs. That is, node u checks whether its data is aggregated with the information sent by any node in ring $i - 1$. If n_{ov} is small, u tries to find a better ring in order to have more of its own data included in the subsequent transmissions. In fact, rings i, $i + 1$, $i - 2$, and $i + 2$ can also be considered for aggregating data (rings $i - 2$ and $i + 2$ could be overheard in case of mobility). To allow for these checks, the list of all node IDs participating in the construction of the synopsis (data aggregation result) is included in the header of each packet. This feature is also exploited at each node as a sort of implicit ACK. Finally, the decision of which ring to join is made according to heuristics depending on n_{i-1}, n_i, n_{i+1}, n_{i+2}, and n_{i-2} [8]. The query aggregation period is divided into epochs, and one aggregate is provided at the end of each. Specific time slots are allocated within each epoch and used to schedule the node transmissions in a TDMA fashion. Sensors can be put to sleep and woken up at their scheduled transmission slots. The aggregation starts from the outermost ring (e.g., R_i), proceeds toward the subsequent ring (e.g., R_{i-1}), and propagates level by level toward the sink. In the example in Fig. 5, the data generated at node A can reach the sink through seven paths: {A, B, F, I, S}, {A, B, F, H, S}, {A, B, F, G, H, S}, {A, C, D, E, I, S}, {A, C, F, H, S}, {A, C, F, I, S}, and {A, C, G, H, S}. Note that, as the main feature of Synopsis Diffusion is that data can flow over multiple paths, a node may receive duplicates of the same information. This may affect the aggregation result, especially when aggregation functions are duplicate sensitive. This problem is addressed by the authors in [8] by proposing proper aggregation functions and data structures. On the upside, multipath schemes are suitable for networks with frequent packet losses due to mobility or channel impairments, as the extra overhead (duplicates) pays off in terms of robustness: if a link fails, the data may still reach the sink through a different path.

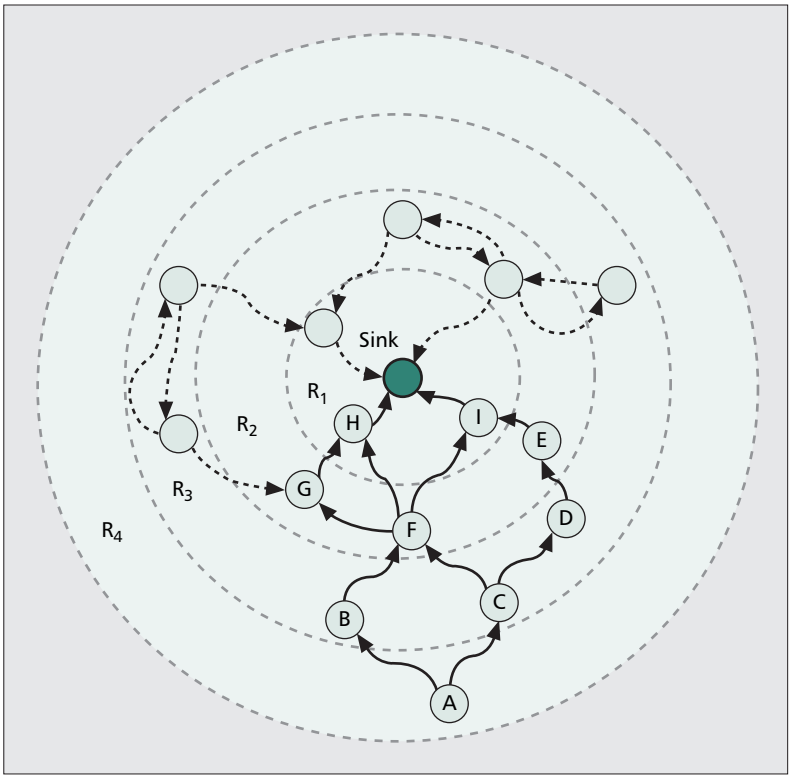


Figure 5. *Examples of aggregation paths over a ring structure.*

Further Algorithms — Another way to implement multipath schemes is based on multiple spanning trees. For instance, the authors in [67] define a method to provide fault tolerance to packet losses by forming a Directed Acyclic Graph (DAG). DAG allows multiple parent nodes at each sensor. In addition, the method ensures correct data transmission timing, according to the actual hop count of the DAG edges.

Hybrid Data Aggregation Approaches

In order to benefit from the advantages of both tree-based and multipath schemes, it is possible to define *hybrid approaches* that adaptively tune their data aggregation structure for optimal performance. To the best of our knowledge, a single work [9] has been proposed with this aim. The related protocol is presented next.

Tributaries and Deltas — The *Tributaries and Deltas* protocol [9] tries to overcome the problems of both tree and multipath structures by combining the best features of both schemes. The result is a hybrid algorithm where both data aggregation structures may simultaneously run in different regions of the network. The idea is that under low packet loss rates, a data aggregation tree is the most suitable structure due to the possibility of implementing efficient sleeping modes (see previous sections) and the good efficiency in representing and compressing the data. On the other hand, in case of high loss rates or when transmitting partial results that are accumulated from many sensor readings, a multipath approach may

In order to benefit from the advantages of both tree-based and multi-path schemes, it is possible to define hybrid approaches which adaptively tune their data aggregation structure for optimal performance.

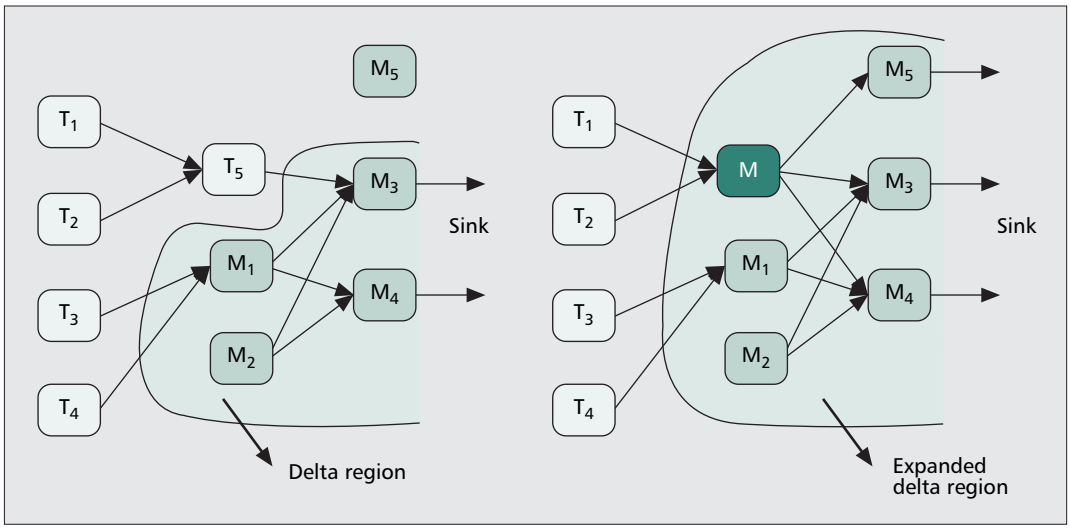


Figure 6. *Example of data gathering regions in Tributary and Delta.*

be the best option due to its increased robustness. Hence, nodes are divided into two categories: nodes using a tree-based approach to forward packets (also called *T* nodes) and nodes using a multipath scheme (*M* nodes). This means that the network is organized in regions implementing one of the two schemes. The main difficulty is to link regions running different data aggregation structures. In doing so, the following rules have to be satisfied [9]:

- *Edge correctness*: An edge originating from an *M* node can never be incident on a *T* node. It means that the aggregation result in a multipath region can only be received by an *M* node (Fig. 6).
- *Path correctness*: *M* nodes form a subgraph including the sink node, which is fed by trees composed of *T* nodes (Fig. 6).

According to the above rules, the sink is surrounded by *M* nodes only. These form the so-called *delta region*, which can be expanded or shrunk by switching nodes from the tree mode (*T*) to the multipath mode (*M*) and vice versa, respectively. In practice, only the nodes lying along the boundary between the two regions are allowed to change their operating mode [9] (Fig. 6). Expanding the delta region corresponds to increasing the number of paths toward the sink, which is good when the packet loss probability is high. On the other hand, shrinking the region is beneficial when the network is static and the packet loss probability is small. The user can set a threshold to specify the minimum percentage of nodes that should contribute to the aggregation operation. Note that this percentage increases in case of a wider delta region. In fact, this implies that more multipath nodes are available, thus leading to higher robustness against failures and, in turn, to more nodes that actively contribute to the aggregation result. The opposite holds when the delta region is shrunk. To see this, consider node T_5 in Fig. 6. This node is switched to an *M* vertex (diagram on the right); as a consequence, it can now also transmit the aggregated data flow to nodes M_4 and M_5. In particular, M_5 can now contribute to the data aggregation by passing the data coming from node *M* to lower levels.

In [9] the authors compare the Tributaries and Deltas algorithm to TAG [5] (pure tree-based) and Synopsis Diffusion [8] (pure multipath). The simulation results in [9] only focus on the quality of the gathered information (root mean square error), while disregarding the energy consumption aspect. In particular, they demonstrate that Tributaries and Deltas guarantees smaller errors with respect to TAG and that the approach nicely solves the drawbacks of pure multipath schemes (Synopsis Diffusion). The major weakness of this approach is the possibly high overhead incurred in updating the data gathering structure. The maintenance of the quite complex network structure may also be a problem with node mobility (this is also an open issue not addressed in [9]).

Finally, particular attention is to be paid to the increase in traffic and therefore to the MAC scheme in use. We stress that *most of the work on data aggregation done so far does not consider this problem*. We emphasize the need for true crosslayer approaches that jointly consider routing, aggregation functions, and MAC aspects with particular focus on both data representation efficiency and energy consumption.

Data Representations and In-Network Aggregation Functions

As discussed earlier, the problems of finding proper data representation and an optimal aggregation function are strongly related and complex. The solutions proposed so far mostly adopt very simple aggregation functions such as average, median, quantile, min, and max [3, 5]. These strongly reduce the amount of data to be transmitted over the network but also heavily affect the precision of the transmitted information (*lossy* aggregation functions). However, in many cases we may be interested in a more detailed representation of the data, which calls for more complex functions and data structures (taking into account the spatial, temporal, and semantic correlation of the readings): cross-layer and self-adaptable data fusion rules have been proposed in [15, 68, 69].

A first improvement to a simple data aggregation function to take into account the spatial correlation is presented in [13]. In this strategy the dependence on the distance among nodes is quantified by a *decay function* which may, for example, decay exponentially with an increasing hop distance [13]. During the data aggregation, each reading is weighed by a decaying factor that decreases with the distance to its source. The framework can be extended by additionally accounting for temporal and semantic correlation. However, this remains an open and mostly unaddressed issue.

In the following sections we describe a selection of in-network aggregation functions according to our classification earlier. We review the simplest methods first, and subsequently consider more complex approaches. At the end of the section, we discuss distributed source coding techniques that perform joint coding of correlated data from multiple sources in a distributed manner.

TiNA

Temporal coherency-aware in-Network Aggregation (TiNA) [14] works on top of a routing tree (i.e., TAG or Cougar) with the data gathering point (sink) as its root. It exploits the temporal correlation in a sequence of sensor readings to reduce energy consumption by suppressing those values that do not affect the expected quality of the aggregated data. This is implemented through a *TOLERANCE* clause added to the SQL query. The *tct* parameter of this clause is used to specify the temporal coherency tolerance for the query. As an example, at a leaf node, each new available value, V_{new}, is compared against the last reported data point, V_{old}. V_{new} is transmitted (and aggregated) up the tree if and only if it satisfies the following requirement (data suppression rule):

$$\frac{|V_{new} - V_{old}|}{V_{new}} > tct. \quad (2)$$

TiNA uses the clause GROUP BY of the SQL query to decide how different messages shall be processed (i.e., two data points can only be aggregated if they belong to the same GROUP). The data gathering procedure executed at the *internal nodes* is as follows. They first gather and combine packets sent by their children. If a given node does not receive valid data from any of its children, it replaces the missing information using the last reported data from the same child (previously stored in its buffer). The node then considers its own reading. If it can be aggregated with some other data in its buffer (they belong to the same GROUP), the reading is aggregated with that data regardless of the *tct* value. Doing so, internal nodes can report their values more often than leaf nodes, thus increasing the accuracy of the aggregation. On the other hand, if the internal node needs to create a new group, it does so and adds the new reading only if this data satisfies Eq. 2. The idea is that new groups are created only when the new measurements significantly differ from old data points (Eq. 2).

Moreover, in TiNA a very simple mechanism to counteract link failures is used. Children, when suppressing data, must send *heartbeat* messages to their parent at regular intervals. The cost of this message is low as it is just a notification packet. Thanks to these packets each parent knows whether its children are still alive. Thus, it can infer whether the old readings are to be kept valid. In case of a missing notification, the appropriate child is discarded until the parent hears from that child again. These messages can also be used in case of mobile sensors as nodes change their location in the network. Finally, the periodic heartbeats allow children to reconnect to the data gathering tree in case of parent failure.

DADMA

Data Aggregation and Dilution by Modulus Addressing (DADMA) [16] is a distributed data aggregation and dilution technique for sensor networks where nodes aggregate or dilute sensed values according to the rules given in an SQL statement. DADMA treats a wireless sensor network as a distributed relational database. This database has a single view that is created by joining records which are locally stored in the sensor nodes. This technique can be used over well-known routing schemes such as Directed Diffusion [1] and LEACH [2]. The sensor network database view (SNDV) is temporarily created and maintained at the sink node. The basic idea in DADMA is to aggregate data coming from a group of sensors or exclude some sensors from the data gathering tree. These operations are carried out according to two simple rules. First, a user can retrieve a subset of data fields available in an SNDV and aggregate data by using the following *aggregate m* function:

$$f_a(x) = x \text{ div } m. \quad (3)$$

Moreover, sensor nodes can be excluded from a query by a *dilute m* function as follows:

$$f_d(x) = (x/r) \bmod (m/r). \quad (4)$$

In the previous equations x is the grid location of a node with respect to one of the axes, r is the resolution in meters, and m is the aggregation (or dilution) factor. As the sink sends a new query, it also specifies a *based on* field and a command that could be either *aggregate* or *dilute*. Each sensor node compares the result of its aggregation or dilution function with the based on value and decides its behavior.

For instance, on receiving a *dilute m* command, a node first uses Eq. 4 to calculate its location indices for both the horizontal and vertical axes ($f_d(x)$ and $f_d(y)$). Subsequently, it compares these values with the x and y indices included in the *based on* field of the query. If they match, the sensor replies to the query. In a similar way, when an *aggregate m* command is received, the values measured by a sensor node are aggregated with those measured by the other nodes having the same indices. We observe that such a strategy is a practical way to take into account the spatial location of the nodes by, for instance, aggregating only those values coming from closely placed devices. The author in [16] studies the performance of DADMA by putting particular emphasis on the energy savings com-

TiNA works on top of a routing tree having the data gathering point (sink) as its root. It exploits the temporal correlation in a sequence of sensor readings to reduce energy consumption by suppressing those values that do not affect the expected quality of the aggregated data.

q-digest is a data structure for representing sensor readings with an arbitrary degree of approximation. The data compression algorithm adapts its behavior to the data distribution by automatically grouping the sensed data into variable size buckets of almost equal weight.

ing from the reduction of the number of transmissions and the probability of event detection. Moreover, he devises a mechanism to achieve a good trade-off among cost, accuracy, and reliability in retrieving the wanted information. The same concepts are addressed in [70] where, in addition to the aggregation/dilution schemes, two location-based hash functions are introduced to determine how the sensed data can be grouped or which sensors should be excluded from a query.

Data Aggregation by means of Feedback Control

The authors of [71] define a strategy to tune the degree of data aggregation while maintaining specified latency bounds on data delivery and minimizing the energy consumption. They consider time-constrained reference scenarios dealing with real-time applications that impose specific time constraints on the delivery of sensor measurements. Data is grouped into different classes associated with different bounds on the delivery time. The aim is to guarantee the delivery of all data *at the minimum energy cost* while *satisfying all time constraints*. The data aggregation degree is adapted accordingly to meet these requirements. If the total communication load exceeds system capacity, the amount of data has to be reduced (the data aggregation degree has to be increased), whereas the data aggregation degree may be relaxed for low traffic. In the former case, a so-called *lossy feedback loop* mechanism assigns a data aggregation degree (d) on the basis of load and capacity estimates. This algorithm runs independently at each node. Specifically, d is defined as the ratio between the number of outgoing and incoming packets. For instance, if $d = 0.66$, three received packets have to be aggregated into two packets (e.g., by averaging two of them). Note that all packets have the same size in this case. In the limiting case where $d = 1$, no data aggregation is performed. Moreover, d is continuously adapted according to new load and capacity estimates. In addition, when the system operates in a non-overloaded regime, a further strategy called *lossless feedback loop* can be used to reduce the energy consumption. According to this scheme incoming messages are collected and transmitted in a single packet without data size reduction.

This solution is interesting for two reasons. First, the control of the data aggregation is based on physical measurements of the network conditions, thus making the mechanism self-adaptable to the actual network dynamics. Second, it aims at satisfying time constraints that, in general, are rarely considered by wireless sensor network algorithms. This solution is extended in [15], where the authors define a complete data aggregation framework (AIDA), by considering general aggregation rules.

Synopsis Diffusion Framework

A recent solution to the data aggregation problem has been proposed in [8]. The main contribution of the article is to define aggregation functions and data structures which are robust to considering the same sensor readings in the data aggregation process multiple times (*double-counting* problem). This is crucial when data aggregation is used in conjunction with multi-path routing schemes.

The approach defines *order and duplicate insensitive* (ODI) properties whose role is to make sure that the final result of the aggregation is independent of the routing topology. That is, the computed aggregate must be the same irrespective of the order in which the sensor readings are merged and the number of times they are considered in the aggregation process. A *synopsis* is defined as a summary of the partial result of the overall aggregation process received at a given node. Three functions on the synopses are possible to perform data aggregation:

- *Synopsis generation*: Given a sensor reading, a synopsis generation function $SG(\cdot)$ produces the corresponding synopsis for that data.
- *Synopsis fusion*: Given two synopses, a synopsis fusion function $SF(\cdot, \cdot)$ generates a new synopsis that summarizes both.
- *Synopsis evaluation*: Given a synopsis, a synopsis evaluation function $SE(\cdot)$ yields up the final result.

The exact implementations of the functions and the synopsis definitions are strictly related to the considered aggregation query. A simple and fast way to check whether a synopsis diffusion algorithm is ODI-correct is based on the following four properties:

- *Preserves duplicates*: if two readings contain the same data values, the algorithm generates the same synopsis.
- The synopsis function $SF(\cdot)$ is *commutative*: for any two synopses $s1$ and $s2$ we have that $SF(s1, s2) = SF(s2, s1)$.
- The synopsis function $SF(\cdot)$ is *associative*: for any triple $(s1, s2, s3)$ we have that $SF(s1, SF(s2, s3)) = SF(SF(s1, s2), s3)$.
- The synopsis function $SF(\cdot)$ is *same-synopsis idempotent*: for any synopsis s, $SF(s, s) = s$.

The four properties above are necessary and sufficient for ODI-correctness. More properties and examples can be found in the related paper [8], where the authors also discuss the advantages of their solution with respect to TAG [5].

The Quantile Digest

Quantile Digest [21] (q-digest) is a data structure for representing sensor readings with an arbitrary degree of approximation (trading data size for precision). The data compression algorithm adapts its behavior to the data distribution by automatically grouping the sensed data into variable size buckets of almost equal weight. As in [21], we assume that sensor readings are integer numbers falling within the range $[1, \sigma]$. A q-digest consists of a set of buckets of different sizes and their associated counts. More specifically, consider a complete binary tree T. In a q-digest, each element of the tree $v \in T$ can be considered as a bucket with a specific range. For example, the range associated with the root of the q-digest is $[1, \sigma]$ and its two children have ranges $[1, \sigma/2]$ and $[\sigma/2 + 1, \sigma]$, respectively. In addition, every bucket $v \in T$ has a counter ($count(v)$) associated with it. The structure is recursive and ranges are halved as we proceed from the root to the leaves of the tree. A q-

digest is simply a subset of the (complete) tree which only contains those elements with positive counts. For its construction, we say that an element of the original tree $v \in T$ is in the q-digest if and only if it satisfies the following properties:

q1) $count(v) \le \lfloor n/k \rfloor$, where n is the number of readings and k is the compression factor. This rule ensures that the internal (non-leaf) element v in the tree does not have a high count.

q2) $count(v) + count(v_p) + count(v_s) > \lfloor n/k \rfloor$ where v_p and v_s are the parent and the sibling of v, respectively.

q3) Since there are no parent and sibling for the root it can violate property q2). A leaf node is instead allowed to violate property q1).

In Fig. 7 we show an example illustrating how a q-digest is built. The example is the same described in [21]. $n = 15$ is the number of readings at any one sensor, which are compressed and summarized in the data structure. The leaf nodes, from left to right, represent the values 1, 2, …, 8 and the number inside the boxes represent the counts. The *compression factor k* is equal to 5, which means that the q-digest has $\lfloor n/k \rfloor = 3$ levels. Finally, $\sigma = 8$ is the size of the data interval, where we assume collecting integer values spanning from 1 to 8. Consider a set of $n = 15$ readings within this range, as shown in Fig. 7a. The number of buckets needed to store all data is 7. In Fig. 7a the children of nodes *a*, *c*, and *d* do not satisfy the digest property q2). Hence, we compress their values into a single bucket by getting to the structure in Fig. 7b. At this point, node *e* still does not satisfy property q2). Hence, we compress the value therein by getting to Fig. 7c. Now, node *g* still does not satisfy property q2); hence, a further compression is needed. This last compression leads us to the q-digest in Fig. 7d. Note that only 5 buckets are needed to store the final result, in spite of the 7 buckets that were originally needed to store the data without compression. As can be observed from this example, this procedure results in a larger loss of accuracy for the readings with a small count. The compression factor k is used to tune the procedure to the desired accuracy. It also affects the memory requirements for storing a q-digest [21].

For its practical implementation, the q-digest structure needs two functions: to construct the q-digest, and to merge two or more q-digests. The first function is called *compress* as it takes the uncompressed q-digest, the number of readings n, and the compression factor k as input, and generates a compressed representation of the q-digest as output (see the above example). The second functionality is the *merge* function, which is used, for example, when two sensors send their q-digests to the same parent. The parent merges these two q-digests into a single q-digest and adds its own values to the new structure. The merge function first takes the union of the two q-digests, which is obtained by adding the counts of the buckets with the same range. After this, it compresses the resulting q-digest by applying the compress function above. As soon as the q-digest structure has been built, each sensor packs it and transmits it to its parent (predecessor node) in the data gathering tree.

In principle, this scheme can be used on top of any routing protocol that avoids loops and duplicates of the same packet. We observe, however, that the joint design of these data representation and compression techniques with routing algorithms is still a completely open research issue.

Distributed Source Coding

A recent paradigm to perform data aggregation exploits *Distributed Source Coding* (DSC). These techniques are based on the Slepian-Wolf theorem [72], which allows joint coding of correlated data from multiple sources and without explicit communication. This is possible as long as the individual source rates satisfy certain constraints about conditional entropies. These techniques require that the correlation structure is available a priori at the independent encoders. Reference [23] gives a good survey on DSC techniques and related open issues in this emerging field. The probably most important contribution to DSC was derived by Slepian and Wolf in their landmark paper [24]. A simple way to encode and transmit the data generated by two generic sources X and Y is to apply separate coding with total rate $R_1 + R_2 = H(X) + H(Y)$, where $H(\cdot)$ denotes the entropy of the data flow. If the two sources can communicate, then they could coordinate their coding operations and use together a total rate of $H(X, Y) \le R_1 + R_2$. The authors in [24] showed that two correlated sources can be coded with a total rate equal to the joint entropy $H(X, Y)$ even though they are *not able* to communicate with each other, as long as their individual rates are at least equal to the conditional entropies $H(X|Y)$ and $H(Y|X)$, respectively. Although different sources do not need to communicate with each other, they *do need* to have some common information about the correlation structure. Toward this end, the sink node may first collect a certain amount of data from the network, process it, and subsequently send the proper correlation information to all sensors. Only after this operation can each node start compressing its readings.

The theory has been generalized and recently applied to wireless sensor networks. For instance, in [73] the authors focus on LDPC codes, which are well known for their capacity of approaching the Shannon limit; Slepian and Wolf proved that the theoretical limit $H(X, Y)$ can be reached with equality, but without devising practical schemes to approach it. In [74], the authors apply Slepian-Wolf coding in its simplest form by proving its effectiveness. Note that, in order for Slepian-Wolf decoding to be effective we need to have a good estimate of data correlation properties. Accordingly, the scheme in [74] uses an algorithm, running at the sink, to measure the actual data correlation. Then, a set of nodes is allowed to send compressed data, where the compression is achieved locally and decoding is performed in a centralized fashion at the data gathering node. At the sink, the uncompressed samples coming from the sensors that are not allowed to compress are used as the side information for decoding. Notably, this approach has the drawback that data is not aggregated along the path to the sink. Hence, further savings can be achieved by exploiting in network data fusion

For its practical implementation, the q-digest structure needs two functions: to construct the q-digest and to merge two or more q-digests. The first function is called compress; the second functionality is the merge function.

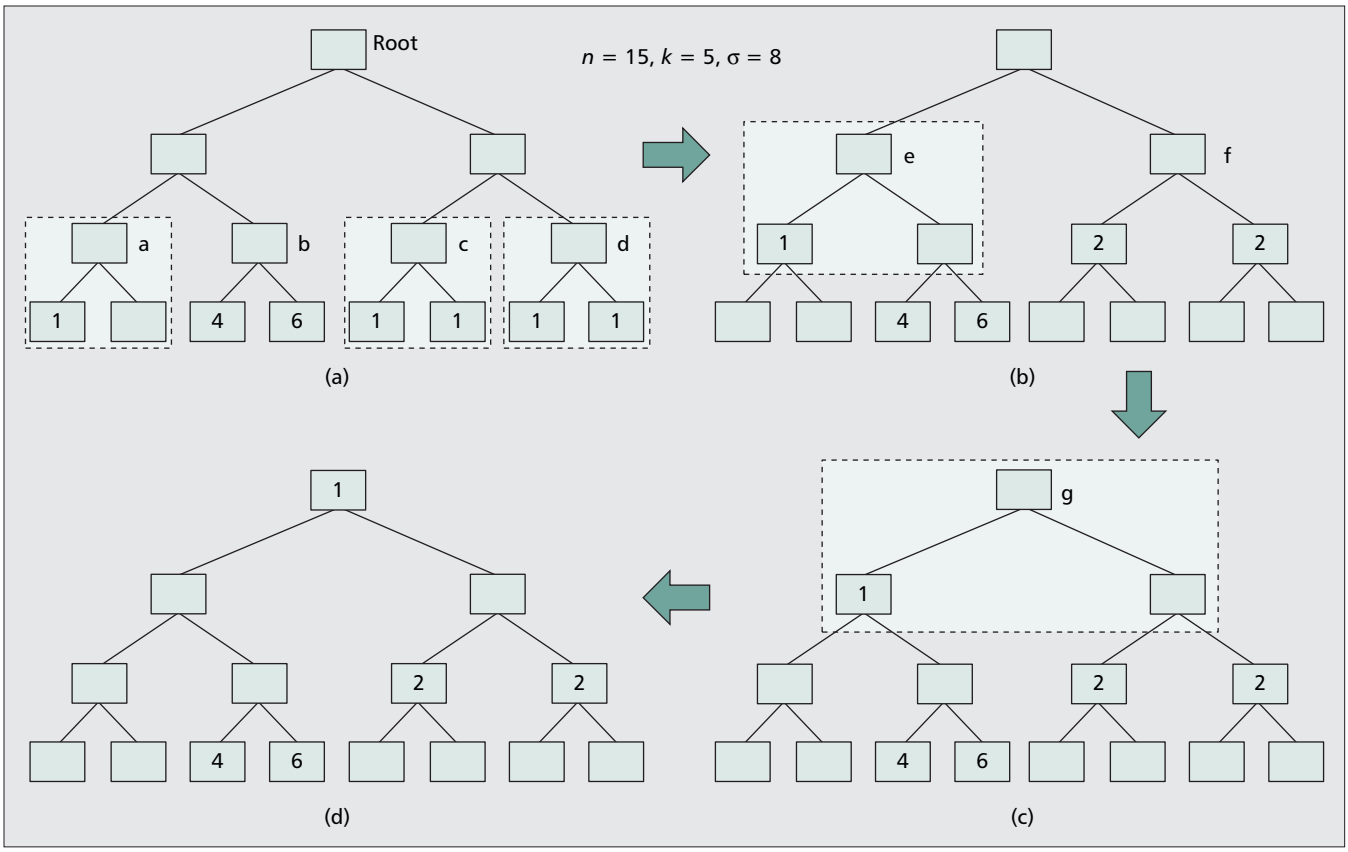


Figure 7. *Q-digest example [21]: the complete tree T is derived by a recursive binary splitting of the original (root) interval [1, σ]. The q-digest consists of the nonempty boxes of the data structure in subfigure (d).*

on top of the distributed per node data compression. Also, this approach might be affected by packet losses as, in such a case, the needed side information might not be fully available at the sink (decoding entity). In the paper, the authors discuss these issues but without giving detailed results. In [75], the authors present and solve the *minimum cost data gathering tree* problem. The network is modeled as a graph $G = (V,E)$, where V and E are the set of vertices (nodes) and edges, respectively. Slepian-Wolf coding is used at every node. Moreover, a communication cost w_e is associated with each edge $e \in E$. The cost function is assumed to be separable, i.e., $f(x_e, w_e) = x_e w_e$, where x_e is the amount of information to be sent over edge e and w_e is the edge cost (e.g., transmission power). The minimum cost data gathering tree problem consists of finding the spanning tree of G and the rate allocation for each node in V that minimize the cost function of the network (i.e., the sum of the costs of all links). The shortest path tree is optimal for any rate allocation and thus the optimization problem can be separated into a spanning tree and a rate allocation optimization subproblem [75]. gives exact algorithms to solve both of them. Overall, the results in [75] allow to code the data in a completely distributed fashion by exploiting the side information in a recursive manner.

The main drawback of this scheme is that it involves the calculation of an SPT, and it requires full (centralized) knowledge of the data correlation structure for all nodes in the network to express the rate constraints. Lossless encoders can then separately and independently encode data at each node as efficiently as if its encoder could see the data values sent by all other nodes. Notably, the scheme's inability to tolerate failures may eliminate this advantage. In fact, if the encoded bits from one node are lost, the sink may not be able to reconstruct several sensor values. The authors of [76] highlight the drawbacks of previous approaches [74, 75] when the network is error prone and, as a partial solution, propose exploiting multipath routing schemes. The advantages of their approach come at the cost of higher energy consumption to set up/maintain multiple trees and transmit multiple copies (extra overhead) of the same message. In summary, DSC effectively makes routing and coding decisions independent of each other. On the downside, however, this solution increases the computational complexity and requires collection of information about joint statistics, which may not always be easy in practice.

Discussions and Conclusions

In this article we have presented a detailed review of in-network aggregation techniques for wireless sensor networks. One of the main design aspects for sensor network architectures is energy efficiency, to keep the network operational as long as possible. Therefore, aggregation techniques are an essential building block, as they aim to reduce the number of transmissions required for data collection, which, in turn, reduces energy consumption.

Algo. / Char.	TAG [5]	Directed Diffusion [1]	PEGASIS [3]	DB-MAC [7]	LEACH [2]	COUGAR [4, 58]	Synopsis Diffusion [8]	Tributaries and Deltas [9]
Aggregation method	Tree-based, online, driven by the sink	Tree-based, online, driven by the sink	Chain-based, centralized or distributed	Completely distributed, asynchronous	Cluster-based, online, distributed	Cluster-based, online, distributed synchronous	Multipath-based, online, distributed	Tree/multipath-based, driven by the sink
Resilience to link failures	Medium	Medium	Low	Medium	Low	Medium	High	High
Overhead to setup/maintain the aggregation structure	High	High	High	Low	Medium	Medium	Medium	Medium
Scalability	Low	Medium	Very Low	High	Low	Low	High	Medium
Resilience in case of node mobility	Low	Medium	Very Low	High	Low	Low	High	Medium
Energy saving methods	Sleeping periods	None	Rotation of the leader	None	Rotation of the cluster head, sleeping periods	Local route repairs	None	None
Timing strategy	Periodic per hop adjusted	Asynchronous	Periodic per hop	Asynchronous	Periodic per hop	Periodic per hop	Asynchronous	Asynchronous

Table 1. *Summary of the basic characteristics of the routing protocols.*

In this survey we have provided a definition of in-network data aggregation and identified its key elements: data dissemination and query mechanisms (with particular focus on routing and MAC layer), aggregation functions, and data structures. Tables 1 and 2 summarize the basic characteristics of the presented solutions and provide a qualitative comparison. By its very nature, in-network aggregation concerns several layers of the protocol stack, and any efficient solution is likely to require a cross-layer design. However, we note that most existing research focuses on networking issues such as routing, often considering only very simple approaches to aggregate data. In addition, much work still remains to be done to provide cross-layer solutions, accounting for application, data representation, routing, and MAC aspects. In fact, the schemes proposed so far often focus on only a subset of these aspects, typically trying to merge routing and data aggregation techniques, but ignoring MAC, application or data representation issues. Finally, another aspect still not deeply investigated concerns the memory and the computational resources necessary to support data aggregation [77].

For routing, many protocols are based on clustering. A major advantage of a clustered structure is that it directly allows aggregation of data at the cluster head. Such algorithms work well in relatively static networks where the cluster structure remains unchanged for a sufficiently long time, but they may be fragile when used in more dynamic environments. Often, the cost required to maintain the hierarchical structure is substantial. Similar considerations apply to tree-based schemes. Initial work addresses some of these problems [53, 78], but further research efforts are required to keep a network functional under mobility. This last aspect is in fact largely unexplored, and it is not clear how different protocols perform in its presence. Also, multipath algorithms may be able to deal with (limited) topology changes due to their higher robustness [8]. An interesting alternative research direction is provided by reactive and localized routing protocols [7]. This study is also one of the very few that take MAC layer issues into account [7, 64]. We stress that without such a joint design, the performance gained at the routing layer may be partially lost due to MAC inefficiencies. Hybrid algorithms allow the combination of the properties of different approaches. This is the case for the algorithm in [9], which provides a good trade-off between tree-based and multipath schemes. Hybrid algorithms allow to tune the degree of aggregation and may facilitate the adaptation of the aggregation scheme (e.g., to the packet loss probability). For these reasons, they are particularly suitable for the design of schemes that are able to adapt to application needs.

As discussed above, only very few studies provide a deeper analysis of the aggregation functions. Previous work mostly takes spatial correlation [13, 79] and temporal correlation [14] of data into account, but semantic correlation is not sufficiently well studied. In this context, distributed source coding is a fairly recent and very promising research area. However, while many theoretical results are known, few of them have been turned into practical algorithms applicable to wireless sensor networks.

References

[1] C. Intanagonwiwat *et al.*, "Directed Diffusion for Wireless Sensor Networking," *IEEE/ACM Trans. Net.*, vol. 11, no. 1, Feb. 2002, pp. 2–16.

[2] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-Specific Protocol Architecture for Wireless Microsensor Networks," *IEEE Trans. Wireless Commun.*, vol. 1, no. 4, Oct. 2002, pp. 660–70.

[3] S. Lindsey, C. Raghavendra, and K. M. Sivalingam, "Data Gathering Algorithms in Sensor Networks using Energy Metrics," *IEEE Trans. Parallel Distrib. Sys.*, vol. 13, no. 9, Sept. 2002, pp. 924–35.

[4] Y. Yao and J. Gehrke, "Query Processing for Sensor Networks," *ACM CIDR 2003*, Asilomar, CA, Jan. 2003.

[5] S. Madden *et al.*, "TAG: a Tiny AGgregation Service for Ad Hoc Sensor Networks," *OSDI 2002*, Boston, MA, Dec. 2002.

For routing, many protocols are based on clustering. A major advantage of a clustered structure is that it directly allows aggregation of data at the cluster head. Such algorithms work well in relatively static networks.

Char. \ Algo.	TiNA [14]	DADMA [16]	AIDA [71]	Synopsis Diffusion [8]	Q-digest [21]
Lossy aggregation	√	√	√/X	√/X	√/X
Duplicate sensitive	X	√	√/X	X	√
Resilience to losses/failures	High	Medium	Low	High	Low
Correlation awareness	√ Temporal	√ Spatial	X	√/X	√ (Temporal, spatial)

Table 2. *Summary of the basic characteristics of the data aggregation functions.*

[6] Y. Xu, J. Heidemann, and D. Estrin, "Geographic-Informed Energy Conservation for Ad Hoc Routing," *ACM/SIGMOBILE MobiCom 2001*, Rome, Italy, July 2001.
[7] G. Di Bacco, T. Melodia, and F. Cuomo, "A MAC Protocol for Delay-Bounded Applications in Wireless Sensor Networks," *Med-Hoc-Net 2004*, Bodrum, Turkey, June 2004.
[8] S. Nath *et al.*, "Synopsis Diffusion for Robust Aggregation in Sensor Networks," *ACM SenSys 2004*, Baltimore, MD, Nov. 2004.
[9] A. Manjhi, S. Nath, and P. B. Gibbons, "Tributaries and Deltas: Efficient and Robust Aggregation in Sensor Network Stream," *ACM SIGMOD 2005*, Baltimore, MD, June 2005.
[10] I. Solis and K. Obraczka, "The Impact of Timing in Data Aggregation for Sensor Networks," *IEEE ICC 2004*, Paris, France, June 2004.
[11] F. Hu, C. Xiaojun, and C. May, "Optimized Scheduling for Data Aggregation in Wireless Sensor Networks," *IEEE ITCC '05*, Las Vegas, NV, Apr. 2005.
[12] X. Jianbo, Z. Siliang, and Q. Fengjiao, "A New In-network Data Aggregation Technology of Wireless Sensor Networks," *IEEE SKG '06*, Guilin, China, Nov. 2006.
[13] E. Cohen and H. Kaplan, "Spatially-Decaying Aggregation Over a Network: Model and Algorithms," *ACM SIGMOD '04*, Paris, France, June 2004.
[14] A. Sharaf *et al.*, "Balancing Energy Efficiency and Quality of Aggregate Data in Sensor Networks," *VLDB J.*, vol. 13, no. 4, Dec. 2004, pp. 384–403.
[15] T. He *et al.*, "AIDA: Adaptive Application-Independent Data Aggregation in Wireless Sensor Networks," *ACM Trans. Embedded Computing Systems*, vol. 3, no. 2, May 2004, pp. 426–57.
[16] E. Cayirci, "Data Aggregation and Dilution by Modulus Addressing in Wireless Sensor Networks," *IEEE Commun. Lett.*, vol. 7, no. 8, Aug. 2003, pp. 355–57.
[17] D. Petrovic *et al.*, "Data Funneling: Routing with Aggregation and Compression for Wireless Sensor Networks," *IEEE SNPA '03*, Anchorage, AK, May 2003.
[18] M. Riedewald, D. P. Agrawal, and A. El Abbadi, "pCube: Updateefficient Online Aggregation with Progressive Feedback and Error Bounds," *IEEE SSDBM 2000*, Berlin, Germany, July 2000.
[19] L. Huang *et al.*, "Probabilistic Data Aggregation in Distributed Networks," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2006-11, Feb. 6, 2006; http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-11.html
[20] X. Wu and Z. Tian, "Optimized Data Fusion in Bandwidth and Energy Constrained Sensor Networks," *IEEE ICASSP '06*, Toulouse, France, May 2006.
[21] N. Shrivastava *et al.*, "Medians and Beyond: New Aggregation Techniques for Sensor Networks," *ACM SenSys '04*, Baltimore, MD, Nov. 2004.
[22] A. Bezenchek, M. Rafanelli, and L. Tininini, "A Data Structure for Representing Aggregate Data," *IEEE SSDBM '96*, Stockholm, Sweden, June 1996.
[23] Z. Xiong, A. D. Liveris, and S. Cheng, "Distributed Source Coding for Sensor Networks," *IEEE Signal Processing*, vol. 21, no. 5, Sept. 2004, pp. 80–94.
[24] D. Slepian and J. K. Wolf, "Noiseless Coding of Correlated Information Sources," *IEEE Trans. Info. Theory*, vol. 19, no. 4, July 1973, pp. 471–80.
[25] R. M. Karp, "Reducibility Among Combinatorial Problems," *Complexity of Computer Computations*, R. Miller and J. W. Thatcher, Eds., Plenum Press, 1972.
[26] B. Krishnamachari, D. Estrin, and S. Wicker, "The Impact of Data Aggregation in Wireless Sensor Networks," *IEEE ICDCS '02*, Vienna, Austria, July 2002.
[27] S. Pattem, B. Krishnamachari, and R. Govindan, "The Impact of Spatial Correlation on Routing with Compression in Wireless Sensor Networks," *ACM/IEEE IPSN '04*, Berkeley, CA, Apr. 2004.
[28] Y. Zhu, K. Sundaresan, and R. Sivakumar, "Practical Limits on Achievable Energy Improvements and Useable Delay Tolerance in Correlation Aware Data Gathering in Wireless Sensor Networks," *IEEE SECON '05*, Santa Clara, CA, Sept. 2005.
[29] Y. Zhu, et al., "A Scalable Correlation Aware Aggregation Strategy for Wireless Sensor Networks," *IEEE WICON '05*, Budapest, Hungary, July 2005.
[30] P. von Rickenbach and R. Wattenhofer, "Gathering Correlated Data in Sensor Networks," *ACM DIALM-POMC '04*, Philadelphia, PA, Oct. 2004.
[31] R. Cristescu *et al.*, "Network Correlated Data Gathering with Explicit Communication: Npcompleteness and Algorithms," submitted to *IEEE/ACM Trans.Net.*
[32] K. Akkaya and M. Younis, "A Survey of Routing Protocols in Wireless Sensor Networks," *Elsevier Ad Hoc Network J.*, vol. 3, no. 3, May 2005, pp. 325–49.
[33] J. N. Al-Karaki and A. E. Kamal, "Routing Techniques in Wireless Sensor Networks: a Survey," *IEEE Wireless Commun.*, vol. 11, no. 6, Dec. 2004, pp. 6–28.
[34] I. Solis and K. Obraczka, "Isolines: Energy-Efficient Mapping in Sensor Networks," *IEEE ISCC '05*, Cartagena, Spain, June 2005.
[35] V. Erramilli, I. Matta, and A. Bestavros, "On the Interaction Between Data Aggregation and Topology Control in Wireless Sensor Networks," *IEEE SECON '04*, Santa Clara, CA, Oct. 2004.
[36] M. Ding, X. Cheng, and G. Xue, "Aggregation Tree Construction in Sensor Networks," *IEEE VTC '03*, Orlando, FL, Oct. 2003.
[37] Y. Yu, B. Krishnamachari, and V. Prasanna, "Energy-Latency Trade-offs for Data Gathering in Wireless Sensor Networks," *IEEE INFOCOM '04*, Hong Kong, Mar. 2004.
[38] H. Luo *et al.*, "Energy Efficient Routing with Adaptive Data Fusion in Sensor Networks," *3rd ACM/SIGMOBILE Wksp. Foundations of Mobile Comp.*, Cologne, Germany, Aug. 2005.
[39] K. Dasgupta, K. Kalpakis, and P. Namjoshi, "An Efficient Clustering-based Heuristic for Data Gathering and Aggregation in Sensor Networks," *IEEE WCNC '03*, New Orleans, LA, Mar. 2003.
[40] H. Albert, R. Kravets, and I. Gupta, "Building Trees Based On Aggregation Efficiency in Sensor Networks," *Med-Hoc-Net 2006*, Lipari, Italy, June 2006.
[41] J. N. Al-Karaki, R. Ul-Mustafa, and A. E. Kamal, "Data Aggregation in Wireless Sensor Networks: Exact and Approximate Algorithms," *IEEE HPSR 2004*, Phoenix, AZ, Apr. 2004.
[42] G. Hartl and B. Li, "infer: A Bayesian Approach towards Energy Efficient Data Collection in Dense Sensor Networks," *IEEE ICDCS 2005*, Columbia, OH, June 2005.
[43] H. O. Tan and I. Korpeoglu, "Power Efficient Data Gathering and Aggregation in Wireless Sensor Networks," *ACM SIGMOD Record*, vol. 32, no. 4, Dec. 2003, pp. 66–71.
[44] H. Cheng, Q. Liu, and X. Jia, "Heuristic Algorithms for Real-Time Data Aggregation in Wireless Sensor Networks," *ACM IWCCC '06,* Vancouver, BC, Canada, July 2006.
[45] Y. Hu, N. Yu, and X. Jia, "Energy Efficient Real-Time Data Aggregation in Wireless Sensor Networks," *ACM IWCCC '06*, Vancouver, BC, Canada, July 2006.
[46] J. Choi *et al.*, "Aggregation Time Control Algorithm for Time constrained Data Delivery in Wireless Sensor Networks," *IEEE VTC 2006-Spring*, Melbourne, Australia, May 2006.

[47] H. Gupta *et al.*, "Efficient Gathering of Correlated Data in Sensor Networks," *ACM MobiHoc 2005*, Urbana-Champaign, IL, May 2005.
[48] B. Zhou *et al.*, "A Hierarchical Scheme for Data Aggregation in Sensor Network," *IEEE ICON '04*, Singapore, Nov. 2004.
[49] C. Intanagonwiwat *et al.*, "Impact of Network Density on Data Aggregation in Wireless Sensor Networks," *IEEE ICDCS '02*, Vienna, Austria, July 2002.
[50] M. Lee and V. W. S. Wong, "An Energy-Aware Spanning Tree Algorithm for Data Aggregation in Wireless Sensor Networks," *IEEE PacRrim 2005*, Victoria, BC, Canada, Aug. 2005.
[51] IEEE LAN MAN Standards, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", *ANSI/IEEE Std.*, Mar. 1999.
[52] R. Kannan and S. S. Iyengar, "Game-Theoretic Models for Reliable Path-Length and Energy-Constrained Routing with Data Aggregation in Wireless Sensor Networks," *IEEE JSAC*, vol. 22, no. 6, Aug. 2004, pp. 1141–50.
[53] H. S. Kim, T. F. Abdelzaher, and W. H. Kwon, "Minimum-Energy Asynchronous Dissemination to Mobile Sinks in Wireless Sensor Networks," *ACM SenSys '03*, Los Angeles, CA, Nov. 2003.
[54] T. Banerjee, K. Chowdhury, and D. P. Agrawal, "Tree based Data Aggregation in Sensor Networks using Polynomial Regression," *IEEE FUSION 2005*, Philadelphia, PA, July 2005.
[55] Y. Yang *et al.*, "SDAP: A Secure HopbyHop Data Aggregation Protocol for Sensor Networks," *ACM MobiHoc '06*, Florence, Italy, May 2006.
[56] Y. Sang *et al.*, "Secure Data Aggregation in Wireless Sensor Networks: A Survey," *PDCAT'06*, Taipei, Taiwan, Dec. 2006.
[57] A. Mahimkar and T. S. Rappaport, "SecureDAV: A Secure Data Aggregation and Verification Protocol for Sensor Networks," *IEEE GLOBECOM 2004*, Dallas, TX, Nov. 2004.
[58] Y. Yao and J. Gehrke, "The Cougar Approach to In-Network Query Processing in Sensor Networks," *ACM SIGMOD Record*, vol. 31, no. 3, Sept. 2002, pp. 9–18.
[59] T. Pham, E. J. Kim, and M. Moh, "On Data Aggregation Quality and Energy Efficiency of Wireless Sensor Network Protocols," *IEEE BroadNets '04*, San José, CA, Oct. 2004.
[60] H. Cam *et al.*, "ESPDA: Energy-Efficient and Secure Pattern-Based Data Aggregation for Wireless Sensor Networks," *IEEE SENSORS '03*, Oct. 2003.
[61] M. Lotfinezhad and B. Liang, "Effect of Partially Correlated Data on Clustering in Wireless Sensor Networks," *IEEE SECON '04*, Santa Clara, CA, Oct. 2004.
[62] O. Younis and S. Fahmy, "Distributed Clustering in Ad-Hoc Sensor Networks: a Hybrid, Energy-Efficient Approach," *IEEE INFOCOM '04*, Hong Kong, Mar. 2004.
[63] V. Mhatre and C. Rosenberg, "Design Guidelines for Wireless Sensor Networks: Communication, Clustering and Aggregation," *Elsevier Ad Hoc Networks J.*, vol. 2, no. 1, Jan. 2004, pp. 45–63.
[64] P. Popovski *et al.*, "MAC-Layer Approach for Cluster-Based Aggregation in Sensor Networks," *IEEE IWWAN '04*, Oulu, Finland, May 2004.
[65] K.-W. Fan, S. Liu, and P. Sinha, "Scalable Data Aggregation for Dynamic Events in Sensor Networks," *ACM SenSys 2006*, Boulder, CO, Oct. 2006.
[66] S. Chen and Z. Zhang, "Localized Algorithm for Aggregate Fairness in Wireless Sensor Networks," *ACM/SIGMOBILE MobiCom 2006*, Los Angeles, CA, Sept. 2006.
[67] S. Motegi, K. Yoshihara, and H. Horiuchi, "DAG Based In-Network Aggregation for Sensor Network Monitoring," *IEEE SAINT '06*, Phoenix, AZ, Jan. 2006.
[68] U. Ramachandran *et al.*, "Dynamic Data Fusion for Future Sensor Networks," *ACM Trans. Sensor Networks*, vol. 2, no. 3, Aug. 2006, pp. 404–43.
[69] M. Wenz and H. Worn, "Event-based Production Rules for Data Aggregation in Wireless Sensor Networks," *IEEE MFI '06*, Heidelberg, Germany, Sep. 2006.
[70] E. Cayirci and T. Coplu, "Distributed Spatial Data Aggregation and Dilution Based on Hashing and Relational Algebra in Wireless Sensor Networks," *IEEE ISSNIP '04*, Melbourne, Australia, Dec. 2004.
[71] T. Abdelzaher, T. He, and J. Stankovic, "Feedback Control of Data Aggregation in Sensor Networks," *IEEE CDC '04*, Atlantis, Paradise Island, Bahamas, Dec. 2004.
[72] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley, 1991.
[73] M. Sartipi and F. Fekri, "Source and Channel Coding in Wireless Sensor networks using LDPC Codes," *IEEE SECON '04*, Santa Clara, CA, Oct. 2004.
[74] J. Chou, D. Petrovic, and K. Ramchandran, "A Distributed and Adaptive Signal Processing Approach to Reducing Energy Consumption in Sensor Networks," *IEEE INFOCOM '03*, San Francisco, CA, Mar. 2003.
[75] R. Cristescu, B. Beferull-Lozano, and M. Vetterli, "On Network Correlated Data Gathering," *IEEE INFOCOM '04*, Hong Kong, Mar. 2004.
[76] S. Coleri and P. Varaiya, "Fault Tolerant and Energy Efficient Routing for Sensor Networks," *IEEE GLOBECOM '04*, Dallas, TX, Nov. 2004.
[77] J. Verd *et al.*, "The Impact of Traffic Aggregation on the Memory Performance of Networking Applications," *ACM MEDEA '04*, Antibes Juan-les-Pins, France, Oct. 2004.
[78] K.-I. Hwang, J. In, and D.-S. Eom, "Distributed Dynamic Shared Tree for Minimum Energy Data Aggregation of Multiple Mobile Sinks in Wireless Sensor Networks," *IEEE EWSN '06*, Zurich, Switzerland, Feb. 2006.
[79] M. Sharifzadeh and C. Shahabi, "Supporting Spatial Aggregation in Sensor Network Database," *ACM GIS '04*, Washington, DC, Nov. 2004.

Distributed source coding is a fairly recent and very promising research area. However, while many theoretical results are known, few of them have been turned into practical algorithms applicable to wireless sensor networks.

BIOGRAPHIES

ELENA FASOLO (fasoloel@dei.unipd.it) received a Laurea degree in telecommunications engineering from the University of Padova in 2004. She is currently a Ph.D. student at the School in Information Engineering at the Information Engineering Department, University of Padova, Italy, and she is spending six months at DoCoMo Euro-Labs in Munich, Germany, to do research on network coding techniques for wireless networks. Her main interests are on networking and MAC protocols especially on efficient data gathering and dissemination schemes.

MICHELE ROSSI [M] (rossi@dei.unipd.it) received both a Laurea degree and a Ph.D. in information engineering from the University of Ferrara, Italy, in 2000 and 2004, respectively. During 2003 he was on leave at the Center for Wireless Communications at the University of California at San Diego. Since November 2005 he is an assistant professor at the Department of Information Engineering of the University of Padova, Italy. His research interests include channel access, routing, and data distribution algorithms for wireless networks.

JÖRG WIDMER [M] (widmer@docomolab-euro.com) is a senior manager at DoCoMo Euro-Labs in Munich, Germany. He leads the Self-Organized Ambient Networking research group, working on several projects in the area of wireless ad-hoc and sensor networks. Before joining DoCoMo, he spent two years as a postdoctoral researcher at EPFL, Switzerland, doing research on ultra-wide band networks. He obtained his M.S. and Ph.D. degrees in computer science from the University of Mannheim, Germany, in 2000 and 2003, respectively. In 1999/2000 he was at the ICSI Center for Internet Research, Berkeley, California, working on equation-based congestion control. His research interests include efficient algorithms for wireless ad hoc and sensor networks, network coding, and transport protocols.

MICHELE ZORZI [F] (zorzi@dei.unipd.it) received a Laurea degree and a Ph.D. in electrical engineering from the University of Padova, Italy, in 1990 and 1994, respectively. During academic year 1992–1993, he was on leave at the University of California, San Diego (UCSD), attending graduate courses and doing research on multiple access in mobile radio networks. In 1993 he joined the faculty of the Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy. After spending three years with the Center for Wireless Communications at UCSD, in 1998 he joined the School of Engineering of the University of Ferrara, Italy, where he became a professor in 2000. Since November 2003 he has been on the faculty at the Information Engineering Department of the University of Padova. His present research interests include performance evaluation in mobile communications systems, random access in mobile radio networks, ad hoc and sensor networks, and energy constrained communications protocols. He is the Editor-In-Chief of *IEEE Wireless Communications*, and currently serves on the Editorial Boards of *IEEE Transactions on Communications*, *IEEE Transactions on Wireless Communications*, *IEEE Transactions on Mobile Computing*, *Wiley Journal of Wireless Communications and Mobile Computing*, and the *ACM/URSI/Kluwer Journal of Wireless Networks*. He was also guest editor for special issues in *IEEE Personal Communications* (Energy Management in Personal Communications Systems) and *IEEE Journal on Selected Areas in Communications* (Multimedia Network Radios).