Efficient Routing from Multiple Sources to Multiple Sinks in Wireless Sensor Networks

Pietro Ciciriello[1], Luca Mottola[1], and Gian Pietro Picco[1,2]

[1] Department of Electronics and Information, Politecnico di Milano, Italy,
{ciciriello,mottola}@elet.polimi.it
[2] Department of Information and Communication Technology, University of Trento, Italy,
picco@dit.unitn.it

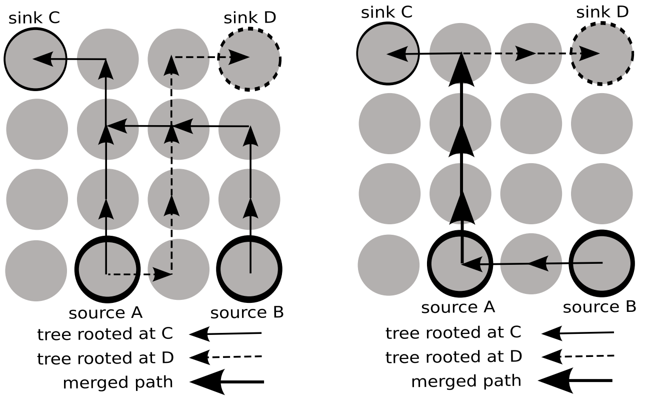
Abstract. Initial deployments of wireless sensor networks (WSNs) were based on a *many-to-one* communication paradigm, where a single sink collects data from a number of data sources. Recently, however, scenarios with multiple sinks are increasingly being proposed, e.g., to deal with actuator nodes or to support high-level programming abstractions. The resulting *many-to-many* communication makes the existing solutions for single-sink scenarios inefficient.

In this paper, we propose a scheme for routing data efficiently from multiple sources to multiple sinks. We first study the problem from a theoretical standpoint, by mapping it to the multi-commodity network design problem. This allows us to derive an optimal solution that, albeit based on global knowledge and therefore impractical, provides us with a theoretical lower bound to evaluate decentralized solutions against. Then, we propose our own decentralized scheme, based on a periodic adaptation of the message routes aimed at minimizing the number of network links exploited. The resulting protocol is simple and easily implementable on WSN devices. The evaluation of our implementation shows that our protocol generates 50% less overhead than the base scheme without adaptation, a result close to the theoretical optimum we derived.

1 Introduction

Early deployments of wireless sensor networks (WSNs) were based on a *many-to-one* paradigm. For instance, in habitat monitoring [1] a single sink node collects environmental data from a large number of sensing devices. Therefore, communication protocols are geared towards the efficient and reliable transmissions to a single receiver.

Recent developments, however, increasingly call for scenarios where the sensed data must be delivered to multiple sinks. This network architecture is obviously required when the same WSN is serving multiple applications, each running on distinct devices. However, the need for multiple sinks arises also in other situations. For instance, researchers are increasingly investigating the use of actuator nodes in WSNs [2]. Different actuators are likely to need data coming from the same set of source nodes, as in the case of an emergency signal and a water sprinkler that cope with a fire scenario by basing their actions on temperature readings sensed nearby. Moreover, multiple sinks are



(a) Two trees rooted at the two sinks are built independently.

(b) Two branches of the trees in Figure 1(a) are merged.

Fig. 1. A sample multi-source to multi-sink scenario.

increasingly and inherently required to implement advanced applications and programming abstractions. For instance, data collection is evolving into complex in-network data mining [3]. In these applications, the mining process is distributed across the nodes in the system, each collecting readings from different sets of data sources. Analogously, to support high-level programming constructs (e.g., the proposals in [4, 5]), the physical nodes in the system need to communicate their data to multiple receivers, where a different processing is performed.

The aforementioned scenarios naturally call for a *many-to-many* communication paradigm. Unfortunately, existing protocols and algorithms for many-to-one communication are inherently ill-suited to cope efficiently with scenarios where the data needs to be reported to multiple sinks. Indeed, available solutions deal with multi-sink scenarios by simply *replicating* the routing infrastructure. For instance, the well-known Directed Diffusion protocol [6] sets up a tree along which sources report their data to the single sink. Dealing with multiple sinks involves setting up a separate, independent tree for each sink—a rather inefficient solution.

To see why this is a problem, consider the sample scenario with two sources and two sinks illustrated in Figure 1(a). Node A reports data to both sinks, whereas node B only transmits to sink C. To achieve multi-hop communication, two trees rooted at the two sinks have been built *independently* (e.g., by flooding a control message from each sink and having each node remember the reverse path to the sink, as in [6]). This base solution exploits 13 networks links and 13 nodes for message routing. Moreover, to report to the two sinks node A is forced to duplicate its data right at the first hop.

Figure 1(b) illustrates a better solution for the same scenario, based on the scheme we describe in the rest of the paper, obtained by maximizing the overlapping between the two sink-rooted trees. The two parallel branches starting from node A have been merged in a single one, and node B leverages off this merged path instead of relying on an independent one. As a consequence, the resulting topology now exploits only 8 network links and 9 nodes. By reducing the number of links exploited, we decrease the amount of redundant information flowing in the network, and duplicate data only if and when strictly necessary. Moreover, less nodes are involved in routing messages. This increases the system life-time, and reduces the contention on the wireless medium and packet collisions, therefore ultimately increasing the reliability of communication. Finally, the readings coming from the two sources can be packed in a single physical message along the merged path, reducing the per-reading header cost.

Our goal in this paper is to support efficiently many-to-many communication from multiple sources to multiple sinks. We do this by enhancing the well-established tree-based solution, thus enabling easy integration of our solution into existing routing schemes, e.g., [6]. Therefore, we assume the presence of a very basic routing infrastructure made of separate trees connecting the sources to the corresponding sinks. In this case, a *single path* connecting a given source to each sink is always established. This is a commonly adopted approach in WSNs, motivated by the reduction in network traffic w.r.t. a solution exploiting multiple paths from a source to the same sink. Furthermore, we do not make any assumption about the pairing of sources and sinks, as it is indeed determined by the initial tree structure. Given this setting, our objective is *to enable efficient routing of messages from the sources to the corresponding sinks by minimizing the number of network links exploited.*

To achieve our goal, we put forth two main contributions:

1. We present a theoretical model of the problem, derived as a particular instance of the multi-commodity network design problem [7, 8]. Thanks to this formulation, we reuse available results and tools for integer programming to easily compute the the theoretical optimal solution to our problem. The model and optimal solution are illustrated in Section 2. The technique we use, however, assumes global knowledge and is therefore derived in an off-line, centralized fashion, impractical for real WSN deployments. Nevertheless, this theoretical result is valuable for providing a lower bound against which to compare more efficient and decentralized solutions.
2. We present and evaluate our own decentralized solution, based on a periodic adaptation of sink-rooted trees. The adaptation consists of selecting a different neighbor as the parent towards a given sink. The decision to adapt is taken locally by a node and is based on the evaluation of a *quality metric* that aggregates into a single value information disseminated by the node's neighbors. Our adaptive protocol, whose details are illustrated in Section 3, is simple enough to be easily implemented on resource-scarce WSN devices. At the same time, as shown in Section 4, the evaluation of our implementation shows that it is able to reduce the network overhead of about 50% w.r.t. the base solution with independent trees, a result close to the theoretical optimum we derive in Section 2.

The paper is concluded by a survey of related efforts in Section 5 and by brief concluding remarks in Section 6.

2 System Model and Optimal Solution

In this section we provide a mathematical characterization of our problem. Besides providing a formal foundation for the results presented in this paper, in this section we show how our model can be used to derive directly an optimal solution, using tools for mathematical programming.

System Model. We can straightforwardly model a WSN as a directed graph whose node set $\mathcal{N}$ is composed of the WSN devices, and whose arc set $\mathcal{A}$ is obtained by setting an arc (i, j) between two nodes i and j when the latter is within the communication range of the former. (Note how this accounts for asymmetric links.)

With this notion of network, the problem of routing from multiple sources to multiple sinks can easily be mapped to the multi-commodity network design problem [7]. In this problem, given a set of commodities $\mathcal{C}$, the goal is to route each *commodity* $k \in \mathcal{C}$ (e.g., a physical good) through a network (e.g., a transportation system) from a set of sources $O(k) \subseteq \mathcal{N}$ to a set of destinations $D(k) \subseteq \mathcal{N}$, by minimizing a given metric. Without loss of generality, as shown in [8], a commodity can be assumed to flow from a single source to a single destination. In this case, since commodities generated from the same source and directed to the same destination follow the same route, one can state a one-to-one mapping between the route connecting any source-sink pair $(o(k), d(k))$, and any commodity k.

Once the mapping to the multi-commodity network design problem is made, we can model our problem as follows:

- We capture message routing with a set of decision variables:

$$r^k_{i,j} = \begin{cases} 1 & \text{if the route for the source-sink pair } k \text{ contains arc } (i, j) \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

  A value assignment $\forall (i, j) \in \mathcal{A}$ to these variables formally represents the route messages must follow from the source $o(k)$ to the sink $d(k)$.
- A network link can be used for multiple source-sink pairs. The fact that an arc (i, j) is used to route *at least one message for a source-sink pair* can then be captured as:

$$u_{i,j} = \begin{cases} 1 & \text{if } \exists k \in \mathcal{C} \mid r^k_{i,j} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

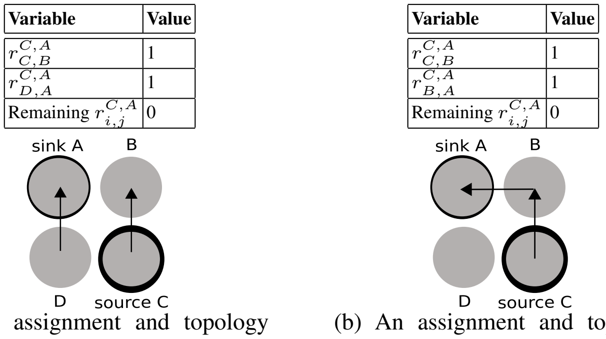
- The overall number of links used to route messages for a given set of source-sink pairs is therefore:

$$\mathit{UsedLinks}(\mathcal{C}, \mathcal{A}) = \sum_{(i,j) \in \mathcal{A}} u_{i,j} \tag{3}$$

Our goal consists of finding the optimal set of routes used to deliver data messages from sources to sinks. Formally:

> *Goal*: to find the value assignment of $r^k_{i,j}, \forall k \in \mathcal{C}, \forall (i, j) \in \mathcal{A}$ that minimizes the value of $\mathit{UsedLinks}(\mathcal{C}, \mathcal{A})$.

The relation between $r^k_{i,j}$ and $u_{i,j}$ defined in (2) captures the essence of the problem, as well as the rationale of our distributed solution, presented next. Indeed, to minimize *UsedLinks* one should strive for reusing as much as possible links that have already been used for other source-sink pairs, i.e., for which the cost $u_{i,j}$ is already paid. In other words, we *can minimize the number of links used by maximizing the overlapping*



Variable	Value
$r_{C,B}^{C,A}$	1
$r_{D,A}^{C,A}$	1
Remaining $r_{i,j}^{C,A}$	0

(a) An assignment and topology representing non-consistent routes for a commodity representing the source-sink path (C, A). Node B and D do not obey to (4).

Variable	Value
$r_{C,B}^{C,A}$	1
$r_{B,A}^{C,A}$	1
Remaining $r_{i,j}^{C,A}$	0

(b) An assignment and topology representing meaningful routes for a commodity representing the source-sink path (C, A). Constraint (4) holds for every node.

Fig. 2. Sample assignments for $r_{i,j}^{C,A}$.

among source-sink paths. In Section 3 we present a protocol for achieving this goal efficiently.

Although this formalization of the problem is simple and general, alternatives exist and are discussed in Section 5.

Finding the Optimal Solution. Based on the model we just presented, we can derive an optimal solution using techniques of mathematical programming, provided that we specify the constraints to be satisfied by a meaningful solution. We first require that $r_{i,j}^k$ and $u_{i,j}$ are integer, binary variables and that the following relation holds among them:

$$\forall (i,j) \in \mathcal{A}, \forall k \in \mathcal{C}, \quad r_{i,j}^k \leq u_{i,j}$$

In our case, these constraints are satisfied by construction through (1) and (2).

Most importantly, we state the requirement that the assignment to $r_{i,j}^k$ contains a connected, end-to-end path for each source-sink pair k. This can be expressed by requiring every node different from the source $o(k)$ and the sink $d(k)$ to "preserve" the message, i.e.:

$$\forall i \in \mathcal{N}, \forall k \in \mathcal{C}, \qquad \sum_{m:(i,m)\in\mathcal{A}} r_{i,m}^k - \sum_{n:(n,i)\in\mathcal{A}} r_{n,i}^k = \begin{cases} 1 & \text{if } i = o(k) \\ -1 & \text{if } i = d(k) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The previous expression is similar to a network flow conservation equation, and indeed imposes the existence of a multi-hop route from each source to every sink. Figure 2 illustrates the concept in the case of a single source-sink pair. The solution in Figure 2(a) is not acceptable, as the message originated at C and directed to A is lost at node B and suddenly reappears at node D. Indeed, the constraint in (4) does not hold for node B and D, as its left-hand side evaluates to -1 when $i = B$ and to 1 for $i = D$, and neither node is an origin or destination for the source-sink pair. Conversely, the solution in Figure 2(b) is perfectly meaningful: a connected, multi-hop path from the source to the sink exists, and indeed the constraint in (4) holds for every node.

With this formulation, the problem of finding the optimal assignment that satisfies our goal can be solved straightforwardly by using well-established techniques and tools from mathematical programming. These techniques require global knowledge of the system state and are computationally expensive, and therefore impractical for WSNs. For this reason, we devised a distributed scheme that relies only on local (i.e., within the 1-hop neighborhood) knowledge, and can be implemented on resource-constrained devices. We return to the theoretical optimal solution in Section 4, where we show how it is efficiently approximated by the distributed solution, discussed next.

3 A Distributed Solution

As we discussed in Section 2, the goal of minimizing the number of links can be achieved by maximizing the overlapping of the paths along which data is routed from a given source to a given sink. In this section, we illustrate the distributed solution we devised to achieve this goal.

We assume that the initial state of the system is such that a tree exists for each sink, connecting it to all the relevant sources. These sink-rooted trees are easily built using mechanisms available in the literature, e.g., along the reverse path of interest propagation as in Directed Diffusion [6]. Clearly, these mechanisms are designed to build each tree independent of the others, and therefore do not guarantee any property regarding their overlapping.

To guarantee a high degree of overlapping among source-sink paths, our protocol relies on a simple adaptive scheme. Each node can decide to locally manipulate a source-sink path by changing the neighbor serving as its parent along a path towards a given sink. The decision is based on information about the neighbor nodes, piggybacked on application messages and overheard during transmission. This control information is fed into a quality metric $q(n, s)$ that yields a measure of the quality of a neighbor n as the parent towards a sink s, and is periodically evaluated for each neighbor n and sink s. Changing the current parent to a different neighbor n occurs when the value of $q(n, s)$ becomes the maximum value of q among all neighbors for sink s. In this case, the node simply begins forwarding data to the new parent. The switch can be managed without additional control messages by using a timeout.

In principle, the quality metric q can be designed to rely on various quantities. In this paper, we present and evaluate an instantiation of our protocol where our quality metric relies on:

1. $dist(j, s)$, the distance (in hops) from a node j to a given sink s, as determined by the initial interest propagation;
2. $paths(j)$, the number of source-sink paths passing through a given node j, i.e., using the notation in Section 2:

$$paths(j) = \sum_{k \in \mathcal{C}} r^k_{i,j} \quad (i, j) \in \mathcal{A}$$

3. $sinks(j)$, the number of sinks a given node j currently serves.

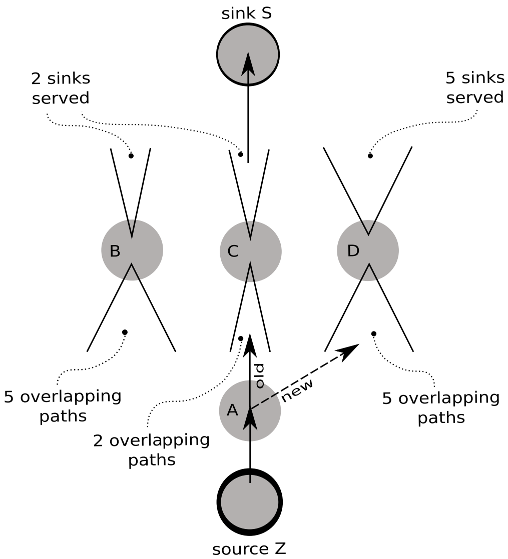


Fig. 3. An abstract view of a WSN with multiple sources and multiple sinks. Source Z generates data to be delivered to sink S, routed through node A. Besides Z, node A is a neighbor of B, C, and D. At node A, the current parent towards S is C. However, a better choice is represented by D, since it enjoys the highest number of overlapping paths and served sinks among A's neighbors.

The distance between a neighbor and a sink is of fundamental importance in increasing reliability and reducing overhead. Indeed, the higher the number of nodes traversed by a message, the higher the probability to lose a message due to unreliable transmission, and the higher the overall computational and communication cost paid to deliver the message end-to-end. The rationale behind the choice of the other two quantities can be visualized with the help of Figure 3. In the network shown, a source Z needs to send data to the sink S, and to do so routes messages upstream through its neighbor A. Node A, in turn, has three neighbors B, C, and D, with C being the current parent in the tree rooted at S. Nevertheless, the figure also shows how both B and D are currently traversed by more source-sink paths than node C. Therefore, if A were to choose either of these neighbors as the new parent towards S, there would be more *overlapping paths* passing through either B or D than in the current situation—which is exactly our goal. Finally, the figure also shows that D is serving more sinks than node B. Therefore, with respect to B, D is more likely[3] to be already reporting readings to S, possibly on behalf of other sources. If this is actually the case, choosing D leads to reusing an "already open" path towards S, therefore further increasing the overlapping of source-sink paths at no additional cost. Therefore, node D is the best choice among A's neighbors, and A will eventually switch to D as its parent towards the sink S.

[3] As we know only the *number* $sinks(j)$ of sinks served by j we cannot be sure that S is really among them. To obviate to the problem, we could propagate the *identifier* of the sinks served instead of their number. However, as shown in Section 4, the latter already yields good performance and generates much less overhead.

Field Name	**Description**
neighborId	The identifier of the neighbor relative to this entry.
dist	An associative array containing, for each sink in the system, its distance from *neighborId*.
paths	The number of different source-sink paths currently passing through *neighborId*.
sinks	The number of sinks served through *neighborId*, possibly along a multi-hop path.

Fig. 4. Information used to compute the quality metric for a neighbor node.

As we already mentioned, the actual decision to switch to a different parent is determined by a quality metric q, an estimate of how "beneficial" this decision would be. In this paper, we designed q to be a linear combination of the three quantities above:

$$q(n, s) ::= \delta \cdot \mathit{dist}(n, s) + \alpha_1 \cdot \mathit{paths}(n) + \alpha_2 \cdot \mathit{sinks}(n) \tag{5}$$

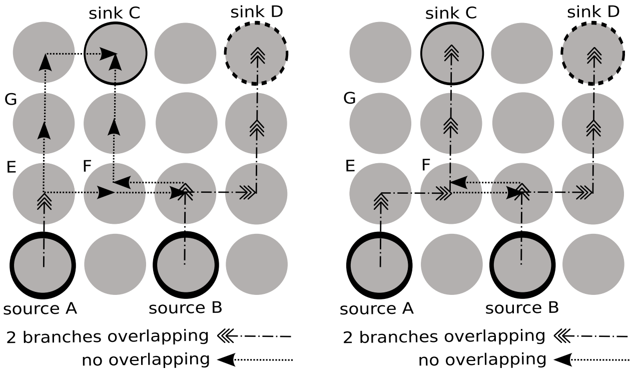
where $\delta, \alpha_1, \alpha_2$ are tuning parameters of the protocol. Again, the shape of the function q and its constituents can in principle be different. For instance, one could take the node remaining energy into account and rely on our solution to automatically alternate among different parents, therefore achieving load balancing among the possible parents in a given tree. Although the results presented in Section 4 with the quality metric in (5) are already very positive, investigating the impact of alternative definitions of q is in our immediate research agenda.

To compute $q(n, s)$ for a given neighbor n and sink s, a node must first determine the three constituents $\mathit{dist}(n, s)$, $\mathit{paths}(n)$, and $\mathit{sinks}(n)$. These are evaluated by relying on a data structure maintained by each node. Figure 4 shows the data structure fields for a single neighbor. Note how the various fields are maintained differently. The value of the field *neighborId* is clearly determined locally based on information from the lower layers, and serves as the key to index the data structure. The content of *dist* is determined from the messages flooded by the sink either during the tree setup phase, or in successive flooding operations performed to keep this information up-to-date with respect to nodes joining or failing. The values of $\mathit{paths}(n)$ and $\mathit{sinks}(n)$ are instead derived by the node through overhearing of messages sent by n. Indeed, these messages piggyback the control information above, which can then be used to update the data structure in Figure 4. Note how the overhead due to this additional control information is very small: only two integer values are needed.

Figure 5 illustrates a sample adaptation process. For the sake of the example, we focus on node E and sink C, and we assume $\delta = \alpha_1 = \alpha_2 = 1$ in (5). With these parameters, node E evaluates the quality metric q towards sink C for its two neighbors F and G. Figure 6 shows the content of the data structures in Figure 4 for F and G. The evaluation returns $q(G, C) = 2 + 1 + 1 = 4$ and $q(F, C) = 2 + 2 + 2 = 6$. Therefore, E recognizes F as the best next-hop towards C, and changes its parent accordingly, as depicted in Figure 5(b). The benefit of this change can be easily seen by computing the number of links and nodes involved: the

Field Name	**Value**
neighborId	**G**
dist	$\{C = 2, D = 4\}$
paths	1
sinks	1
neighborId	**F**
dist	$\{C = 2, D = 4\}$
paths	2
sinks	2

Fig. 6. Data stored at node E in the situation depicted in Figure 5(a).



(a) Initial configuration.

(b) Node E changes its parent from G to F.

Fig. 5. A sample adaptation process.

network in Figure 5(a) uses 13 network links and 12 nodes, against the 10 links and 10 nodes of Figure 5(b).

To break ties between the current parent and a new one, a node always selects the latter, as it is guaranteed to enjoy a higher value of q after becoming a parent. Indeed, at least the number of source-sink paths passing through it increases by one. In selecting the new parent, the only additional constraint is to not select as a new parent a neighbor whose distance from a sink is greater than that of the selecting node. Without this constraint, a node could potentially select one of its children as the new parent, hence creating a routing loop.

Finally, our distributed protocol is complemented by a simple scheme for packing multiple readings in the same network message. To this end, each node maintains a buffer for each neighbor, limited by the number of readings allowed in a message. Upon receiving a reading from another node, the reading is inserted in the buffer for the neighbor on the route to the target sink. When the buffer for a given neighbor is full (or upon expiration of a timeout) a message is created and actually forwarded to the neighbor. This simple scheme decreases the per-reading header cost and helps reducing collisions, since buffers are likely to become full at different times and therefore messages are going to be reasonably spread in time. In principle, the same packing scheme can be used without our adaptation protocol. However, its impact is greater in the presence of adaptation, since the latter guarantees a higher degree of overlapping among trees, with more readings being funneled through the same links.

4 Evaluation

In this section, we report about simulation results comparing the performance of our solution against a base mechanism without adaptation as well as the optimal solution identified in Section 2. These solutions provide the two extremes for our evaluation: we indeed demonstrate that our adaptation strategy provides remarkable benefits, and that its effectiveness approaches the theoretical optimum. In our evaluation, we also show that our solution converges rapidly, and investigate the impact of the various constituents of our quality metric q, as introduced in Section 3. We are currently investigating alternative definitions of q (e.g., including the remaining node energy for achieving load balancing, as mentioned earlier).

Simulation Settings and Metrics. We implemented our distributed scheme on top of TinyOS [9], and evaluated its performance using the TOSSIM [10] simulator. As for the theoretical optimum discussed in Section 2, we used the CPLEX [11] solver to compute the ideal topology connecting sources to sinks, given their respective placement in the system and the constraints defined in Section 2.

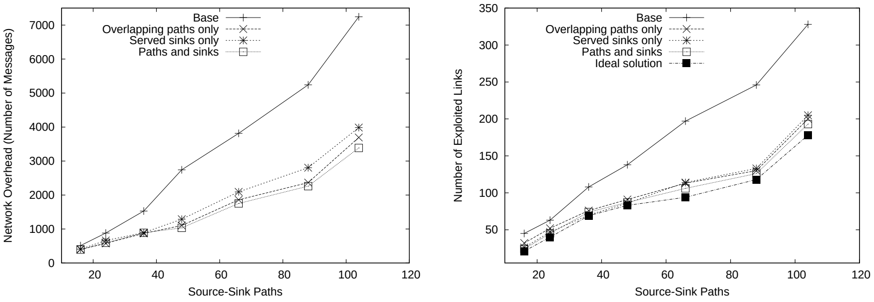
We first report about simulated deployments in a regular grid, where each node can communicate with its four neighbors. This choice simplifies the interpretation of results by removing the bias induced by random deployments, while also well modeling some of the settings we target, e.g., indoor WSN deployments for control and monitoring [12]. To this end, the nodes are placed 35 ft. apart with a communication range[4] of 50 ft. Moreover, we also evaluated the performance of our protocol in deployments with a random topology, characterized by a pre-specified average number of neighbors for each node. With respect to the fixed grid above, these scenarios allow us to assess the impact of the connectivity degree on our results, as well as evaluate our protocol in more unstructured scenarios (e.g., modeling outdoor WSNs deployments).

As for the modeling of sources and sinks, each scenario is set so that 10% of the nodes are data sources. These send data to a number of sinks that varies according to the scenario, at the rate of one reading per minute. Hereafter, the time period between two successive readings generated from the same source is termed *epoch*. Also, note that sources are not synchronized in generating these readings. The placement of sources and sinks in the network is determined randomly. A single sensor reading is represented by a 16-bit integer value, while the message size at the MAC layer is 46 bytes. In all our implementations, messages are always sent when there are sufficiently many readings to fill the physical message completely. Each simulation lasted 2000 s, and was repeated 5 times.

The initial tree is built by flooding the system with a "tree construction" message sent by every sink. Each node keeps track of the messages received from the same sink, and stores the identifier of the neighbor along which the message was received with the least number of traversed hops. This way, the initial tree is built by minimizing the length of the path connecting each source to every sink. In the chart, this *base* tree is also used, without any additional modification, as the point of comparison against our solution. Indeed, it essentially provides a baseline, representative of protocols that build independent trees (e.g., Directed Diffusion [6]), against which we show the benefits of our adaptive scheme.

For what concerns the protocol parameters, we evaluate independently the impact of the number of overlapping paths and the number of served sinks by simulating scenarios with $\langle \alpha_1 = 1, \alpha_2 = 0 \rangle$ and $\langle \alpha_1 = 0, \alpha_2 = 1 \rangle$. Moreover, we also evaluate the combined contribution of these quantities using $\langle \alpha_1 = 1, \alpha_2 = 1 \rangle$. As for the distance

[4] We used TinyOS' `LossyBuilder` to generate topology files with transmission error probabilities taken from real testbeds.



(a) Network overhead (forwarded messages).

(b) Number of links exploited.

Fig. 7. Grid topology: performance metrics vs. number of source-sink paths.

from sinks, we discussed in Section 3 how its contribution is key in achieving a good ratio of delivered readings. Differently from the two quantities above, the lower is this value, the better (closer) is the neighbor located w.r.t. a given sink. For this reason, we always set $\delta = -2$ throughout all the simulation runs, so that neighbors at a few hops from the considered sink are preferred over neighbors farther away. As we verified experimentally, this value provides a good trade-off w.r.t. the other parameters.

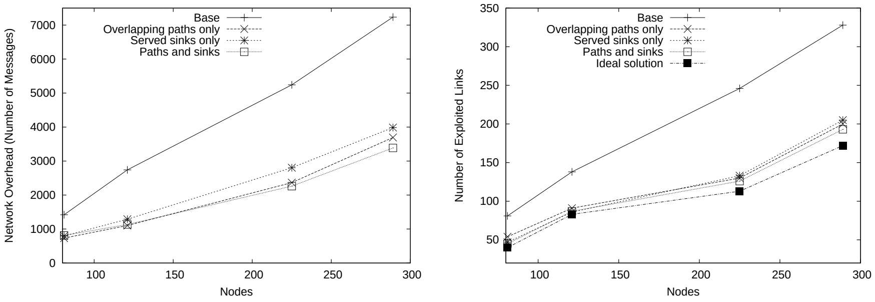
The main quantities we measured are:

- the *ratio of readings delivered* to the sinks over those sent;
- the *network overhead* as the number of messages sent at the MAC layer—being communication the most prominent source of energy drain in WSNs, this measure can be considered as proportional to the system lifetime;
- the *number of links exploited*, i.e., the number of physical links used to route messages—the fundamental metric we strive to minimize[5].

Moreover, to provide further insights on the behavior of our protocol we analyzed the number τ of trees insisting on each physical link, showing the ratio $\tau_{adaptive}/\tau_{base}$ in different scenarios. Finally, to analyze the dynamics of our protocol we measured the *number of topological changes* against the epoch number, showing the time needed for our solution to stabilize.

Results. We first focus on a grid topology. The comparison of the charts in Figure 7 captures the essence of our approach. Figure 7(a) plots the network overhead against the number of source-sink paths, and shows how our adaptive scheme exhibits only about 50% the overhead of the base solution, on the average. On the other hand, Figure 7(b) shows that our distributed scheme relies on only about 50% of the links used by the base solution. Remarkably, the number of network links exploited in Figure 7(b) exhibits the same trend of the overhead in Figure 7(a), therefore evidencing that the gains in network overhead are made possible by the reduction in the number of links exploited. Indeed, in our approach messages are duplicated only where it is really necessary, whereas the base solution often duplicates messages too early, as they are routed independently. Furthermore, Figure 7(b) shows also a curve for the theoretical optimum we computed

[5] Notice we count multiple links also when different sinks can be reached with a single broadcast message at the physical level.



(a) Network overhead (forwarded messages).

(b) Number of links exploited.

Fig. 8. Grid topology: performance metrics vs. number of nodes (4 sinks).

in Section 2. Remarkably, our solution always achieves a performance very close to the optimum—at most 10% in the worst case—but *without* requiring global knowledge. Note how these significant improvements in overhead are obtained without impacting message delivery: actually, the ratio of delivered readings improves of about 10% in our adaptive scheme[6].

Figure 7 also evidences that considering only the number of served sinks in our quality metric q yields the worst results in the adaptive solution—although still significantly better than the base scheme. The combination of the two metrics provides the best results in scenarios with a high number of source-sink paths, exhibiting a gain around 10% w.r.t. the number of overlapping paths alone. Conversely, little or no improvement is obtained by $\langle \alpha_1 = 1, \alpha_2 = 1 \rangle$ in settings with less sources and sinks. A closer look at our simulation logs revealed that in these scenarios the combination of overlapping paths and served sinks simply amplifies the differences in the value of q for different neighbors, only seldom changing the decision on the parent to be selected. This is partially expected, as in the aforementioned settings more overlapping paths easily correspond to more served sinks and vice versa.

Thus far, we analyzed the performance of our protocol only w.r.t. the number of source-sink paths. Indeed, we noted the performance of our approach is affected more directly by this parameter than by the number of nodes in the system. For instance, we obtained comparable performance in a scenario with 81 nodes (8 sources) and 4 sinks, w.r.t. a setting with 121 nodes (12 sources) and 3 sinks. The settings shown thus far were obtained with different system sizes, starting from 2 sinks in a system of 81 nodes, up to 4 sinks among 289 nodes. Nevertheless, to investigate the scalability properties of our solution, Figure 8 illustrates the same trends discussed above, this time against the number of nodes in the system in a scenario with 4 sinks. The adaptive scheme scales fairly well w.r.t. system size and, again, much better than the base solution with no adaptation, and very close to the theoretical optimum. Moreover, once more the trend of network overhead (Figure 8(a)) is mirrored by the one for the number of exploited links (Figure 8(b)).

[6] Due to space limitations we do not show the corresponding charts here.

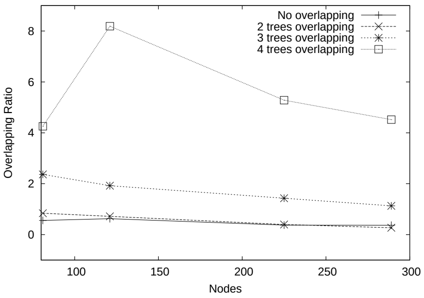


Fig. 9. Ratio of overlapping source-sink paths per physical link vs. number of nodes.

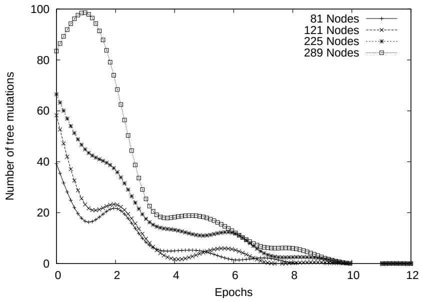
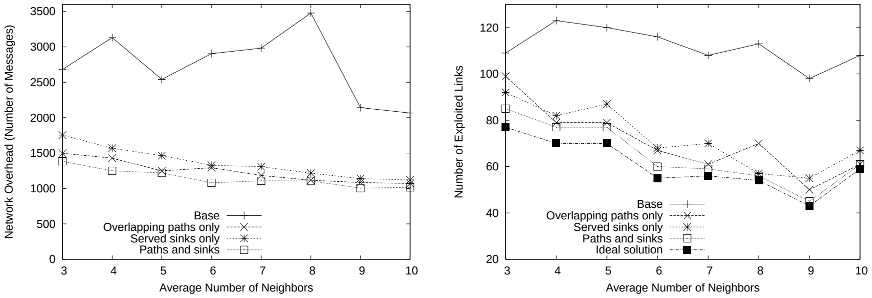


Fig. 11. Convergence time, in a system with 4 sinks.



(a) Network overhead (forwarded messages).

(b) Number of links exploited.

Fig. 10. Random topology: performance metrics vs. average number of neighbors per node.

A finer-grained analysis is shown Figure 9, where we show the ratio $\tau_{adaptive}/\tau_{base}$ of overlapping trees per physical link using $\langle \alpha_1 = 1, \alpha_2 = 1 \rangle$. As expected, based on the previous considerations about network overhead and number of exploited links, our adaptive scheme sensibly increments the number of source-sink paths overlapping on the same link. For instance, in a scenario with 121 nodes, the number of links shared among 4 trees is about 8 times the one for the base scheme. However, as the number of nodes increases with a fixed number of sinks, this ratio decreases since the different source-sink paths may become too far from each other to be merged effectively.

The impact of connectivity is considered in Figure 10, with a random deployment of the nodes. The charts show the network overhead and number of used links against the average number of neighbors per node in a system with 150 nodes, 4 of which are sinks. The trends are more irregular in this case, because of the bias introduced by the random topology. However, Figure 10(a) shows that our solution always outperforms the base scheme, achieving improvements from 40% to 60% in network overhead. As for the number of physical links exploited, shown in Figure 10(b), it is interesting to notice that, as connectivity increases, the gap between our solution and the theoretical optimum is reduced. Indeed, the more the system is globally connected, the more information the nodes collect by overhearing messages, and the more options they enjoy when selecting a parent for a given tree. Our distributed scheme should converge to the optimal solution as the system becomes more and more connected. Incidentally, the trend in Figure 10(b) also highlights how our previous choice of 4 neighbors per node in the grid topology was fairly conservative.

Finally, as our solution is based on successive rounds of adaptation, we also analyzed the time needed to converge to a stable configuration. Figure 11 shows the number of topological changes against the epoch number. Results are obtained in network of various sizes with 4 sinks. The chart shows that the higher is the number of nodes in the system, the higher is the number of topological changes and the time needed to converge. However, in the worst case 10 epochs are sufficient to reach a stable configuration, with most of the changes concentrated in the first few epochs. Therefore, in stable deployment scenarios there is no need to run continuously the adaptation process, which instead can be triggered with a large period or only upon detecting topology changes.

5 Related Work

The model we presented in Section 2 is derived from the large body of literature in operational research and network design. Our choice of the multi-commodity network design problem as a modeling framework is motivated by the generality it allows in pairing sources and sinks. In contrast, modeling the same problem as a p-source minimum routing cost spanning tree [13] or a Steiner minimal tree [14] would force us to consider *every* node (or source, respectively) to be a sink as well. At the same time, the model we presented here is a simple instance of the multi-commodity network design problem. More sophisticated formulations exist, e.g., taking into account the capacity of network links [15]. In this case, when the capacities along a path are exhausted, alternative, parallel paths are used to share the traffic load, therefore activating more links. However, in WSNs it is difficult to evaluate precisely the actual bandwidth available, due to contention of the wireless medium, collisions and unreliable transmissions [16]. Moreover, these issues are amplified as the number of links used to route messages increase. Therefore, we believe these formulations are not suited for the wireless setting.

For what concerns distributed solutions, it is safe to say that most research work in sensor network focuses on optimizing communication from multiple sources to a single sink, as witnessed by the vast amount of literature on the subject [17]. As we already mentioned, these approaches cannot provide efficient solutions to more decentralized scenarios like sensor and actuator networks [2], which inherently call for routing solutions to report to multiple receivers.

In [18] the authors propose mechanisms to build sink-rooted trees incrementally, to perform data aggregation and in-network processing. A path from a single source to the sink is first built, and then shared by other, nearby sources. In this sense, their approach resembles our rational of minimizing the number of network links exploited to reduce the network traffic. However, their solution is geared to single-sink scenarios, and the results barely comparable to ours, as they are obtained in simulation using a MAC layer derived from IEEE 802.11. Devising mechanisms to combine the two techniques could provide further benefits, and is a topic worth further investigation.

The work in [19] addresses the problem of routing from a single source to multiple sinks. Common to our approach is the use of broadcast transmissions to let nodes collect information on alternative routes. However, the adaptation in [19] is performed based on *long-range* information (e.g., the number of hops from a node to the different sinks). As this information may not be immediately available, the algorithm starts with a worst case estimation and randomly tries different routes, including those deemed less favorable. When the information gathered during this exploration phase is not modified for a given number of iterations, the algorithm switches to a stable phase where the discovered routes are used. The adaptation mechanism we proposed in this work is instead based mainly on *local* information that is immediately available (e.g., the number of source-sink paths passing through a node), Moreover, our algorithm is basically self-stabilizing, and does not require distinct phases of operation.

Some researchers addressed the problem of routing from multiple sensors to mobile sinks, focusing on mechanisms to deal with frequent location updates. To this end, in [20] a two-level grid structure is proactively built by the sources. This identifies a reduced subset of nodes responsible for storing information about the sink position, and to which location updates are sent. Conversely, in [21] a stationary sensor node builds a tree on behalf of one or more mobile sinks. These remain linked to this node until they move too far away, at which point they are forced to select a different stationary node. In-network data processing in the presence of mobile sinks is also considered in [22], where a tree is built by a master sink and then shared by slave sinks. Local repair strategies are employed to adjust the tree according to sink mobility. Differently from our approach, in these works sink mobility is the distinctive feature of the target scenario, and the proposed solutions are aimed at reducing the overhead induced by it. In contrast, we concentrate on optimizing the source-sink paths, as this is key to improve the system lifetime in our target scenarios, actually less dynamic. In doing so we make only minimal assumptions about the node capabilities (i.e., the ability to overhear messages sent by neighbors), while all the aforementioned proposals require nodes to be aware of their geographical position, exploited for routing.

Instead, the work in [23] introduces an algorithm targeting monitoring applications for achieving energy-efficient routing to multiple sinks. The optimizations proposed are centered around the ability to adjust the sensing rate at different nodes, eliminating the redundancy in the data gathered while preserving the ability to reconstruct the corresponding phenomenon. Instead, we do not assume the ability to influence the source behaviors. Conversely, common to our approach is the problem formulation based on integer linear programming. The authors then map this formulation to a distributed search algorithm based on subgradient optimization, executed in a decentralized fashion. However, they do not provide any insights on the processing overhead this solution would impose on real, resource constrained nodes. We use instead the model presented in Section 2 as a theoretical bound for careful analysis of a lightweight, distributed solution straightforwardly implementable on WSN devices.

Finally, other works have focused on the opportunity to employ multiple sinks not to meet an application requirement, but as a mechanism to increase the system lifetime. For instance, the work in [24] investigates the design problem related to optimally locating multiple sinks in the sensor field, so as to achieve a pre-specified operational time. In this case, even if multiple sinks are present, these simply act as cluster-heads, with each sensor node reporting to only one of them. Similarly, the proposal in [25] studies the problem of selecting, at each node, one of the many sinks present in the system to minimize the overall energy expenditures. Clearly, this is a different problem w.r.t. ours, where the multiple sinks actually represent different system actors, that need to *simultaneously* gather sensor data for potentially different tasks.

6 Conclusions and Future Work

This paper addressed the problem of efficiently routing data from multiple sources to multiple sinks. We defined a model based on mathematical programming that, albeit relying on global knowledge, can be easily used to derive an optimal solution. Then, we illustrated a novel decentralized scheme that adapts the topology by maximizing the overlapping among source-sink paths, therefore minimizing the overall number of network links exploited. This results in reduced overhead, increases the system lifetime, and makes communication more reliable. The approach is simple enough to be implemented on resource-scarce WSN devices, and we showed that achieves a 50% improvement in network overhead. Additionally, we illustrated how the quality of the routes obtained is close to the theoretical lower bound.

Our immediate research agenda includes an assessment of the impact of our adaptive strategy on the processing load at each node, and the design of techniques for balancing such load using alternative definitions of the quality metric we introduced in this paper. A formal proof of the convergence properties of our distributed scheme is also among our future research goals.

Acknowledgements. The authors wish to thank Roberto Cordone and Marco Trubian for their insightful suggestions on modeling the multiple sources to multiple sinks routing problem, and Kay Römer for providing suggestions and comments on early versions of this paper. The work described here is partially supported by the European Union under the IST-004536 RUNES project.

References

1. Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., Anderson, J.: Wireless sensor networks for habitat monitoring. In: Proc. of the 1^{st} ACM Int. Workshop on Wireless sensor networks and applications. (2002) 88–97
2. Akyildiz, I.F., Kasimoglu, I.H.: Wireless sensor and actor networks: Research challenges. Ad Hoc Networks Journal **2**(4) (2004) 351–367
3. Roemer, K.: Distributed mining of spatio-temporal event patterns in sensor networks. In: Proc. of the 1^{st} Euro-American Wkshp. on Middleware for Sensor Networks (EAWMS). (2006)
4. Bakshi, A., Prasanna, V.K., Reich, J., Larner, D.: The abstract task graph: a methodology for architecture-independent programming of networked sensor systems. In: Proc. of the 2005 Wkshp. on End-to-end, sense-and-respond systems, applications and services (EESR), Berkeley, CA, USA, USENIX Association (2005) 19–24
5. Ciciriello, P., Mottola, L., Picco, G.P.: Building Virtual Sensors and Actuator over Logical Neighborhoods. In: Proc. of the 1^{st} ACM Int. Wkshp. on Middleware for Sensor Networks (MidSens06 - colocated with ACM/USENIX Middleware). (2006)

6. Intanagonwiwat, C., Govindan, R., Estrin, D., Heidemann, J., Silva, F.: Directed diffusion for wireless sensor networking. IEEE/ACM Trans. Networking **11**(1) (2003) 2–16
7. Wu, B.Y., Chao, K.M.: Spanning Trees and Optimization Problems. Chapman & Hall (2004)
8. Holmberg, K., Hellstrand, J.: Solving the uncapacitated network design problem by a lagrangean heuristic and branch-and-bound. Oper. Res. **46**(2) (1998) 247–259
9. Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., Pister, K.: System architecture directions for networked sensors. In: ASPLOS-IX: Proc. of the 9^{nt} Int. Conf. on Architectural Support for Programming Languages and Operating Systems. (2000) 93–104
10. Levis, P., Lee, N., Welsh, M., Culler, D.: TOSSIM: accurate and scalable simulation of entire TinyOS applications. In: Proc. of the 5^{th} Symp. on Operating Systems Design and Implementation (OSDI). (2002) 131–146
11. I-Log: CPLEX Home Page. (`www.ilog.com/products/cplex/`)
12. Stoleru, R., J.A. Stankovic: Probability grid: A location estimation scheme for wireless sensor networks. In: Proc. of the 1^{st} Int. Conf. on Sensor and Ad-Hoc Communication and Networks (SECON). (2004)
13. Wu, B.Y., Lancia, G., Bafna, V., Chao, K.M., Ravi, R., Tang, C.Y.: A polynomial-time approximation scheme for minimum routing cost spanning trees. SIAM J. Comput. **29**(3) (2000) 761–778
14. Hwang, F.K., Richards, D.S., Winter, P.: The Steiner Tree Problem. North-Holland (1992)
15. Gendron, B., Crainic, T.G., Frangioni, A.: Multicommodity capacitated network design. Telecommunications Network Planning (1998) 1–19
16. Pottie, G.J., Kaiser, W.J.: Wireless integrated network sensors. Commun. ACM **43**(5) (2000) 51–58
17. Al-Karaki, J., Kamal, A.E.: Routing techiniques in wireless sensor networks: a survey. (To appear in IEEE Wireless Communications)
18. Intanagonwiwat, C., Estrin, D., Govindan, R., Heidemann, J.: Impact of network density on data aggregation in wireless sensor networks. In: Proc. of the $22th$ Int. Conf. on Distributed Computing Systems (ICDCS), Washington, DC, USA, IEEE Computer Society (2002) 457
19. Egorova-Förster, A., Murphy, A.L.: A Feedback Enhanced Learning Approach for Routing in WSN. Technical report, University of Lugano (2006) Available at `www.inf.unisi.ch/phd/foerster/publications/foerster06.pdf`.
20. Luo, H., Ye, F., Cheng, J., Lu, S., Zhang, L.: Ttdd: Two-tier data dissemination in large-scale wireless sensor networks. Wireless Networks (11) (2005) 161–175
21. Kim, H.S., Abdelzaher, T.F., Kwon, W.H.: Minimum-energy asynchronous dissemination to mobile sinks in wireless sensor networks. In: Proc. of the 1^{st} Int. Conf. on Embedded networked sensor systems (SENSYS). (2003) 193–204
22. Hwang, K., In, J., Eom, D.: Distributed dynamic shared tree for minimum energy data aggregation of multiple mobile sinks in wireless sensor networks. In: Proc. of 3^{rd} European Wkshp. on Wireless Sensor Networks (EWSN). (2006)
23. Yuen, K., Li, B., Liang, B.: Distributed data gathering in multi-sink sensor networks with correlated sources. In: Proc. of 5^{th} Int. IFIP-TC6 Networking Conf. (2006) 868–879
24. Oyman, E.I., Ersoy, C.: Multiple sink network design problem in large scale wireless sensor networks. In: Proc. of 1^{st} Int. Conf. on Communications (ICC). (2004)
25. Das, A., Dutta, D.: Data acquisition in multiple-sink sensor networks. Mobile Computing and Communications Review **9**(3) (2005) 82–85