Routing protocol for anycast communications in a wireless sensor network

Nancy El Rachkidy, Alexandre Guitton, Michel Misson
{nancy,guitton,misson}@sancy.univ-bpclermont.fr

LIMOS-CNRS, Clermont University
Complexe scientifique des Cézeaux,
63177 Aubière cedex, France

Abstract. In wireless sensor networks, there is usually a sink which gathers data from the battery-powered sensor nodes. As sensor nodes around the sink consume their energy faster than the other nodes, several sinks have to be deployed to increase the network lifetime. In this paper, we motivate the need of anycast communications in wireless networks, where all the sinks are identical and can gather data from any source. To reduce interference and congestion areas on the wireless medium, the path from a source to a sink has to be distant from the path connecting another source to another sink. We show that determining distant paths from sources to sinks is an NP-hard problem, and we propose a linear formulation in order to obtain optimal solutions. Then, we propose a sink selection and routing protocol called S4 and based on realistic assumptions and we evaluate it through simulations. Finally, we conclude that anycast routing protocols in wireless sensor networks should not compute paths independently for each source, but rather consider all the sources simultaneously.

Keywords: Sink selection and routing protocol, anycast communications, wireless sensor networks.

1 Introduction

In the past few years, wireless sensor networks have been used in several monitoring applications. Battery-powered nodes collect data with their sensors, and send it in a wireless manner to a data-gathering station, called the sink. The sink usually has a large memory capacity, and sometimes does not even have energy limitations, contrarily to the sensor nodes. The sink has several roles: it can store historical data, analyze the data to detect discrepancies or emergency situations, or act as a gateway providing connectivity with a wired network.

As the traffic of the network converges to the sink, nodes close to the sink consume their energy faster than farther nodes. When all the nodes around the sink have depleted their energy, the sink is not able to receive any data from the sensors, and gets disconnected from the network. When this situation happens, the whole network is considered to be down. A solution to this problem is to deploy several sinks. If traffic is balanced among the sinks, the network lifetime can be significantly increased since the energy consumption will be almost equal for all the nodes in the network.

The paradigm of anycast communications, also termed one-to-any communications, becomes very important in a network with multiple sinks: when a sensor node produces data, it has to send it to any sink available. A sink selection strategy is to choose for each source a sink arbitrarily. An alternative strategy is to route data to the closest sink. Assuming that the sources and the sinks are uniformly distributed in the network, this simple strategy is assumed to balance the energy consumption.

In this paper, we show that in order to minimize interference and to reduce the congestion areas between the different paths, all the sources have to be considered simultaneously. Figure 1 shows a topology with two sources s_1 and s_2 and two sinks d_1 and d_2, with three sink selection strategies, in order to give an insight of why considering sources simultaneously is important. On part (a) of the figure, each source is connected to the closest sink, which generates contention around sink d_2. On part (b), each source is connected to a different sink in order to balance the sink load. However, the path (s_1, d_2) intersects with the path (s_2, d_1). Contention on the wireless medium is generated around the area where the paths meet. On part (c), paths (s_1, d_1) and (s_2, d_2) are distant from each other. The wireless traffic generated on one path has little impact on the traffic generated on the other path (provided that those paths are distant enough). This third sink selection strategy can only be achieved by considering all the sources and sinks simultaneously. Also note that the sink selection and the routing have to be performed at the same time.

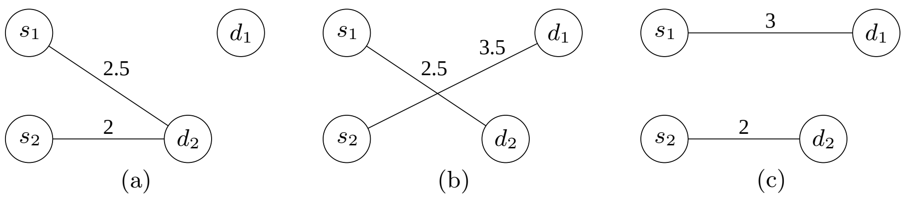

Fig. 1. To minimize interference and reduce congestion on the paths from each source to a sink, all the sources and sinks have to be considered simultaneously, as in (c).

The paper is organized as follows. Section 2 describes the related works. Section 3 formally describes the problem of finding distant paths for a set of anycast communications. We prove that the problem is NP-hard, and we propose an integer linear formulation to obtain optimal solutions. Section 4 describes our strategy, based on pivot routing in order to ensure that paths are distant. Section 5 describes our simulation environment and settings, and provides the simulation results we obtained. Finally, we conclude our work in Sect. 6.

2 Related work

In this section, we focus on three main topics: the deployment of multiple sinks, the anycast paradigm which allows sources to send data to any sink, and the use of multipath in wireless routing protocols.

2.1 Multi-sink deployment

Multi-sink deployment refers to a wireless network architecture where several sinks are deployed, each of them having identical functional capabilities. Recently, interest is emerging towards scenarios with multiple sinks in order to improve the network lifetime and to ensure a fair delivery of data among sinks [1,2]. Another advantage of deploying multiple sinks is to improve the data gathering by reducing the communication delay from sensors to sinks [3,4] or the total communication cost [5].

2.2 Anycast communications

Anycast is a one-to-any communication paradigm where a source communicates with a single sink, chosen among a set of possible sinks. It has been shown that the lifetime of a wireless sensor network can be increased by deploying several sinks, and accessing them using an anycast protocol [6,7]. In [8], the authors also showed that anycast forwarding schemes can significantly reduce the expected packet delivery delays.

2.3 Multipath routing

Multipath routing is a feature that enables a source to send packets to a destination through multiple different paths at the same time. The main advantage of this feature is to improve the reliability of the packet delivery: even if one path becomes blocked due to a node failure for instance, the destination is still able to receive packets as long as at least one path is active. Using multipath also helps balancing the energy consumption among the network and therefore extends network lifetime. Most of the multipath routing protocols are based on classic on-demand single path routing methods [9,10].

Disjoint multipath routing methods try to determine disjoint paths, *i.e.*, paths that do not have nodes or edges in common. As stated in [11], using disjoint multipaths does not remove the potential for collisions, resulting in large packet loss rates and reduced data transmission performance. The main reason is that wireless transmissions might interfere communications between distant nodes. In [12], the authors aim to find zone-disjoint multipaths using directional antennas. A promising approach is described in [13], where authors proposed an energy efficient and collision aware disjoint multipath routing algorithm. The flooding required to determine such paths is limited to nodes close to the main discovery route.

In this paper, we do not use multipath to route traffic from a source to a sink. We rather aim to determine a collection of paths (one per source-sink pair) that are distant from each other. However, as shown in the next section, the problem of finding disjoint multipaths between a source and a sink, and between two source-sinks pairs are closely related.

In [14], the authors proposed a probabilistic proactive routing protocol called PiRAT and based on pivots. When an emergency situation occurs, several geographically close sensors might produce alarm messages that have to be forwarded to a sink. The paths followed by all the alarms becomes congested and several alarm packets might be dropped. By selecting randomly distant pivot nodes for each source, PiRAT is able to reduce the congestion areas and to improve the network performance in terms of delay and packet loss. Indeed, it allows diversity in routing and avoids congested areas of the network, which contributes to balance the traffic load and the energy consumption between nodes. In Sect. 4, we use a similar pivot approach in order to obtain distant paths.

3 Problem statement and modelling

In this section, we study the problem of determining a sink for each source. We also study the related problem of finding a path from each source to its assigned sink, so that the paths from all the sources are distant from each other. We focus on providing a formal description of the sink selection and routing strategy for anycast communications.

Let us consider a set of sensors V forming a wireless sensor network $G = (V, E)$. E is defined in the following way: if x and y can communicate with each other, we have $(x, y) \in E$ and $(y, x) \in E$. Let $S \subset V$ denote a set of sources, and $D \subset V$ denote a set of sinks. Finally, let us denote by $h(x, y)$ the hop count between two nodes x and y. The minimum distance between two paths p_1 and p_2, denoted by $h(p_1, p_2)$, can be defined as:

$$h(p_1, p_2) = \min_{x \in p_1, y \in p_2} h(x, y).$$

Definition 1. *The sink selection and routing problem for anycast wireless communications (SSRPAW) consists in finding a set of paths $\{p_i\}$ that connects each source $s_i \in S$ to a sink $d_{f(i)} \in D$, such that $h(p_{i_1}, p_{i_2}) \geq \delta$ for any $i_1 \neq i_2$ and for a given $\delta > 0$.*

The rationale behind the SSRPAW problem is to find a sink selection strategy (characterized by the function f) and a routing strategy (characterized by the choice of paths $\{p_i\}$) that ensures that paths are distant enough from each other to avoid contention in the medium. The number of hops between two different paths is at least δ. δ depends on the propagation conditions. In a dense network, interference is often negligible after two hops, and thus δ is often 2.

In the remainder of this section, we show that SSRPAW is NP-hard. Then, we propose an integer linear program that allows to compute optimal solutions. Finally, we show by simulations that only small instances have optimal solutions.

This motivates the need of a heuristic that can work with limited computational capabilities and realistic assumptions.

3.1 Proof of the NP-completeness of SSRPAW

In order to prove the NP-completeness of SSRPAW, we first have to define a similar problem.

Definition 2. *The set-to-set disjoint path problem takes as input a graph $G = (V, E)$, a set of k sources S and a set of k destinations D. It consists in determining if there are k mutually node-disjoint paths $\{p_i\}$, such that p_i is a path from s_i to d_{j_i}, for $1 \leq i \leq k$ and j a permutation of $\{1, \dots, k\}$. Two mutually node-disjoint paths have no node in common, except for the source and the destination.*

Theorem 1. *The set-to-set disjoint path problem is NP-complete [15]. It is similar to the node-to-node disjoint path problem.*

Theorem 2. *SSRPAW is NP-complete for any δ.*

Proof. The proof of the NP-completeness of SSRPAW for any $\delta > 0$ is by reduction to the set-to-set disjoint path problem. We show in the following that if one is able to solve SSRPAW on a specific graph $\bar{G}$ in polynomial time, one has solved the set-to-set disjoint path problem in a general graph G in polynomial time (which is unlikely, unless $P = NP$).

Let $G = (V, E)$ be an arbitrary general graph. The construction of $\bar{G} = (\bar{V}, \bar{E})$ is the following. Each node $n \in V$ is also a node of $\bar{V}$. Each edge $e = (x, y) \in E$ becomes a path of δ edges in $\bar{E}$, connecting $x \in \bar{V}$ to $y \in \bar{V}$.

Let us now assume that SSRPAW can be solved in polynomial time in a graph $\bar{G}$. This means that there are $|S| = k$ paths $\{\bar{p}_i\}$ in $\bar{G}$, for $1 \leq i \leq k$, such that each path $\bar{p}_i$ connects a source $s_i \in S$ to a sink $d_{j_i} \in D$. Moreover, for any $i_1 \neq i_2$, $h_{\bar{G}}(\bar{p}_{i_1}, p_{i_2}) \geq \delta$, by definition of SSRPAW. By construction of $\bar{G}$, each path $\bar{p}$ in $\bar{G}$ can be translated into a path p in G. Thus, we have k paths $\{p_i\}$ in G such that each path p_i connects a source $s_i \in S$ to a sink $d_{j_i} \in D$. These paths are such that $h_G(p_{i_1}, p_{i_2}) \geq \delta/\delta = 1$, which means that they are node disjoint. Thus, we have solved the set-to-set disjoint path problem between S and D in polynomial time, which completes the proof.

3.2 Integer linear formulation

The goal of this subsection is to define optimal solutions by an integer linear program. This program takes as input a set of nodes V, a set of sources $S \subset V$, a set of sinks $D \subset V$ and a set of binary variables $e_{x,y}$ representing the edges. The objective of the integer linear program is to find a set of paths $\{p_s\}$, one per source $s \in S$ and to any sink $d \in D$, such that paths are distant from each other and the total number of edges used is minimized. Each path is defined as a minimize $\sum_{s\in S}\sum_{x\in V}\sum_{y\in V} p_s(x,y)$

such that $\forall s \in S, x \in V, y \in V \qquad p_s(x,y) \leq e_{x,y}$ (1)

$\forall s \in S \qquad \sum_{y\in V} p_s(s,y) \geq 1$ (2)

$\forall s \in S \qquad \sum_{x\in V}\sum_{d\in D} p_s(x,d) \geq 1$ (3)

$\forall s \in S, x \in V \backslash D, y \in V \backslash D$

$$p_s(x,y) \leq \sum_{z\in V\backslash\{x\}} p_s(y,z) \qquad (4)$$

$\forall x \in V \qquad \sum_{s\in S}\sum_{y\in V} p_s(x,y) \leq 1$ (5)

$\forall s_1 \in S, s_2 \in S\backslash\{s_1\}, x \in V, x' \in V$

$$\sum_{y\in V} p_{s_1}(x,y) + \sum_{y\in V} p_{s_2}(x,y) \leq \#S - (\#S-1)e_{x,x'} \qquad (6)$$

same as previous with $p_{s_1}(x,y)$ and $p_{s_2}(x',y)$ (7)
same as previous with $p_{s_1}(x',y)$ and $p_{s_2}(x,y)$ (8)
same as previous with $p_{s_1}(x',y)$ and $p_{s_2}(x',y)$ (9)

Table 1. Integer linear constraints.

set of binary variables $p_s(x,y)$, such that $p_s(x,y)$ is 1 if path p_s uses edge (x,y), and 0 otherwise. The resulting objective function is given on Table 1.

The problem constraints are given on Table 1. Inequality (1) states that it is forbidden to use an edge (x,y) in a path if x and y are not neighbors (or equivalently, if there is no edge (x,y) in the graph, that is if $e_{x,y} = 0$). Constraint (2) indicates that for any source s, there is at least one edge that leaves s in p_s. Constraint (3), symmetrically, indicates that each path p_s should terminate in a node of D. This constraint corresponds to anycast communications. Constraint (4) is a connectivity constraint: it states that for each edge (x,y) of a path p_s (except for sinks), there is at least one edge (y,z) (with $z \neq x$). In other words, it says that each edge (x,y) on a path p_s is followed by an edge (y,z) on the same path. Constraint (5) states that each edge (x,y) is used by at most one path. Constraints (6), (7), (8) and (9) indicate that the distance between paths should be of at least two or more. Thus, the program only[1] applies to $\delta = 2$. More specifically, it says that:
(i) If an edge (x,x') exists in the graph, there is at most one path that uses node x or x', since x and x' are neighbors. Indeed, if the edge (x,x') exists, $e_{x,x'}$ is equal to 1, and the right part of the equation is equal to $\#S - (\#S-1) = 1$ (where $\#S$ represents the cardinal of S).
(ii) If the edge (x,x') does not exist in the graph, the number of paths that uses nodes x and x' is not limited. In this case, $e_{x,x'} = 0$ and the right part of the equation is equal to the maximum number of paths $\#S$. As $\#S$ is a natural limit

[1] A similar approach can be applied for larger values of δ, but drastically increases the number of constraints.

to the number of paths, the inequality does not bring restriction in this case. These four equations cannot be merged into one, because it is possible for a single path p_{s_1} to use the edge (x, x'). In this case, the summation would count two edges for this path (one is (x, x') and the other is (x', y)), and the result would not anymore be smaller than 1. Note however that Constraint (5) (corresponding to $\delta = 1$) is not required when $\delta > 1$, as it is included in Constraints (6) to (9).

3.3 Computation of optimal solutions

In this subsection, we study the optimal solutions found by our integer linear program, using the GLPK (GNU Linear Programming Kit) solver. We generated a grid topology of 7×7 nodes, such that each node can communicate to its four direct neighbors only. We tried to find paths from each source to any sink, with a minimum distance of $\delta = 2$ between paths. Sources and sinks are chosen randomly such that all the sources are at a distance of δ of each other, and all the sinks are at a distance of δ of each other. We varied the number of sources and the number of sinks. Results are averaged over 50 simulations.

Figure 2 shows that the average optimal path cost increases with the number of sources and sinks. When the number of sources and sinks becomes larger, the paths are longer in order to ensure disjoint paths. Figure 3 shows the percentage of optimal paths found as a function of the number of sources and sinks. When the number of sources is smaller than four, optimal solutions are always found. When the number of sources is five or six, there are no optimal solutions. This means that it is not possible to find distant paths with $\delta = 2$. The given integer linear formulation is too restrictive, and is not applicable in a realistic setup.

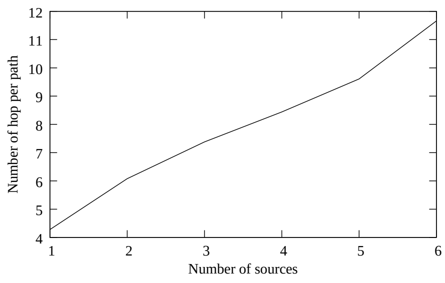

Fig. 2. Average optimal path cost per source-sink pair as a function of the number of sources, when the number of sinks is equal to the number of sources.

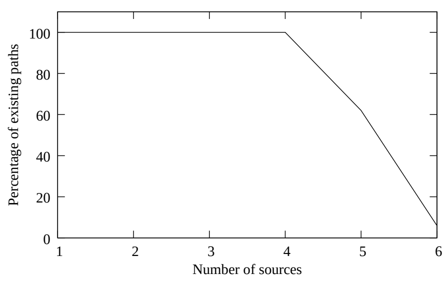

Fig. 3. Percentage of topologies having an optimal solution as a function of the number of sources, with the number of sinks equal to the number of sources.

Simulations are run on a standard personal computer and they took about two hours and 35 minutes. The results show that even without limited computational capabilities, optimal solutions cannot always be found, even on small instances. In a wireless sensor network, where nodes have very limited computational capabilities, there is a strong need for a simple heuristic that is able to provide approximated solutions using realistic assumptions.

4 Heuristic sink selection strategies

As we have shown in the previous section, it is not realistic to find optimal disjoint paths from sources to sinks. Moreover, the optimal formulation makes assumptions about the network that are not realistic. This is why we propose in this section an heuristical approach called simultaneous sink selection strategy, combined with pivot routing. Before describing our strategy, we present two commonly used strategies.

Random sink selection (RSS): In RSS, sinks are randomly chosen. Packets are routed using the shortest path from the source to the selected sink. We expect this strategy to perform badly in terms of packet loss, as it might incur congestion in several areas of the networks. Congestion also increases the delay due to retransmission attempts.

Closest sink selection strategy (CSSS): In CSSS, each source is connected to the closest sink. The distance from the source to each sink can be computed according to the geographical coordinates or using hop count. CSSS does not take into account the fact that the areas around sinks can be congested. It uses the shortest path between the source and the selected sink.

Simultaneous sink selection strategy (S4): S4 uses a greedy algorithm to select sinks and to compute paths: each source is considered sequentially. For each source, a pivot node is selected in order to make paths as disjoint as possible. The pivot selection works as follows. When considering a source s, S4 considers all the nodes as potential pivots and all the sinks as potential destinations. For each potential pivot x and potential destination d, S4 determines the hop count $h(s, x)$ between s and x, and the hop count $h(x, d)$ between x and d. S4 also determines the number of nodes in common between the path $p(s, x, d)$ from s to d via x and $\mathcal{P}$, the union of all the paths already chosen by S4 for the previous sources. For a given source s, S4 has several candidate paths. S4 chooses one of the paths that minimizes the number of nodes in common with $\mathcal{P}$. If there are still several candidate paths, S4 chooses one of the paths of minimum length among the candidates. Notice that S4 combines a sink selection strategy with a routing mechanism. Thus, S4 reduces the energy consumption since it balances the traffic in the whole network.

The hop count between two nodes can be computed using a simple signalling protocol, or using properties of hierarchical addresses (such as those used in IEEE 802.15.4 for example). This process is out of scope of this paper. The computation of the number of nodes in common between two paths can be computed by having s sending a first message to x, and a second message via x

to d. Each node on both path can send a notification back to s, which is then able to count the number of common nodes. Another possibility is again to use properties of hierarchical addresses, which allows a node s to determine the path between two nodes a and b, provided that the addresses of a and b are known. This process is also out of scope of this paper. We assume here that s has a centralized knowledge of the topology.

In order to operate S4, we consider that there is a centralized entity which knows the whole topology (as mentioned previously) and computes paths for all the sources. For each source s, this entity has to compute the shortest paths from s to any node, and the shortest paths from any sink to any node. The first computation requires $\mathcal{O}(n+m)$ operations, where n is the number of nodes and m is the number of edges, and the second requires $\mathcal{O}(n+m)$ too (note that a single computation is performed for all the sinks). Then, all the nodes are considered as pivots and all the sinks are considered as destinations. Thus, the overall complexity is $\mathcal{O}(|S|(n+m)|D|)$. As $|D|$ and $|S|$ are supposed to be rather small, the burden of the centralized entity in S4 is reasonable.

5 Evaluation

In this section, we describe the simulations we ran in order to compare S4 with RSS and CSSS. We used the NS-2 simulator, version 2.31. We used the IEEE 802.15.4 physical and MAC layers with the non-beacon enabled mode. The propagation model used was the two ray ground model, with default parameters. The transmission power was set to a realistic value of -25 dBm, and the radio range was set to 25 m. The size of the nodes queue was set to 50 packets.

In our simulations, we considered for simplicity reasons a set of 49 nodes uniformly distributed on a grid of 70×70 square meters. Each node is located at a distance of 10 m of its neighbors. All sensors were full function devices with routing capabilities. The PAN coordinator was located at the center of the area. We waited for the network to be fully associated before injecting data packets. Data packets of 77 bytes (at the physical layer) were generated during 50 seconds, at a fixed rate of 2 packets per second. The routing protocol we used for the three strategies was AODV[2] [16]. Notice that before AODV can send packets to an unknown destination, it has to establish a path through reply and request messages, which introduces delay[3]. In order to have stable results for RSS, results are averaged over 500 repetitions.

5.1 Performance metrics

Our algorithm is evaluated and compared to RSS and CSSS (detailed in Sect.4), according to two performance metrics:

[2] AODV was slightly modified in S4 in order to allow pivot nodes

[3] For RSS and CSSS, AODV has to establish paths from each source to its assigned destination. For S4, AODV has to establish paths from each source to its pivot, and from this pivot to the sink.

(i) Packet loss: the packet loss is defined as the number of packets received by the sink nodes over the number of packets generated by the source nodes. Thus, the packet loss metric takes into account the losses due to collisions and the losses due to a large number of retransmission attempts.
(ii) End-to-end delay: the end-to-end delay is the time interval between the transmission of a packet by the source and the reception of the same packet by the sink, at the application layer. The time required by AODV to establish routes, as well as the delay introduced by retransmissions, are taken into account into the end-to-end delay. However, the end-to-end delay only takes into account the packets that are correctly received by the sink.

5.2 Packet loss

Figure 4 and Fig. 5 show the average packet loss as a function of the number of sources and sinks respectively. We notice that the packet loss for the three strategies increases consistently with the number of sources, and decreases with the number of sinks. When the number of sources in the network is large, the traffic load is large too and the medium is overloaded by the generated packets. S4 is able to significantly reduce the packet loss compared to RSS and CSSS. Indeed, S4 aims to build distant paths by selecting pivots for each source-sink pair, which contributes to balance the traffic between nodes and to reduce congestion on the medium. S4 reduces by approximately 36% the packet loss probability of RSS and CSSS, for five sources and five sinks. S4 outperforms the other two strategies when the number of sinks is one: S4 reduces by approximately 41% the packet loss probability of RSS and CSSS for five sources and one sink.

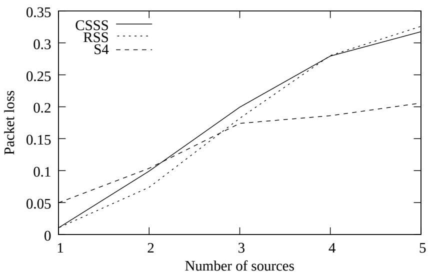

Fig. 4. Packet loss probability as a function of the number of sources, with as many sinks as sources.

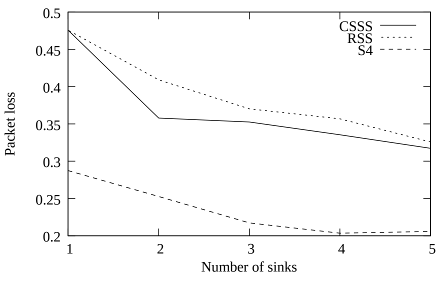

Fig. 5. Packet loss probability with five sources, as a function of the number of sinks.

5.3 End-to-end delay

Figure 6 shows the average end-to-end delay for the three strategies as a function of the number of sources. For RSS, the delay increases with the number of sources and sinks and becomes stable when there are more than four sources in the network. The delay is positively affected by the number of sinks (as the average distance between a source and a sink decreases) and negatively affected by the traffic load (as the medium becomes congested). It can be noticed that CSSS induces larger delays than RSS. This is explained by the fact that CSSS tends to select the same sink for several close sources, which yields to congested areas around the sinks. RSS balances the sink usage by choosing sinks randomly. While CSSS and RSS have almost the same packet loss, the impact on delay is significant: packets with large delay are more likely to be dropped in RSS, which reduces the average packet loss for this strategy. S4 has the best behavior of the three strategies. This proves that it is important to consider all sources simultaneously for the sink selection, and to build distant paths during the routing process. S4 reduces the average end-to-end delay over CSSS by 70% and over RSS by 50%, for five sources and five sinks.

Figure 7 shows the average end-to-end delay for the three strategies, for five sources, as a function of the number of sinks. For the three strategies, we notice that the delay decreases with the number of sinks. With a large number of sinks, there are less congested areas in the network, and thus the number of packet retransmissions decreases (because the packet loss decreases too, see Fig. 5). S4 outperforms the two other strategies, even with one sink: in this case, it reduces the end-to-end delay of CSSS and RSS by 47%.

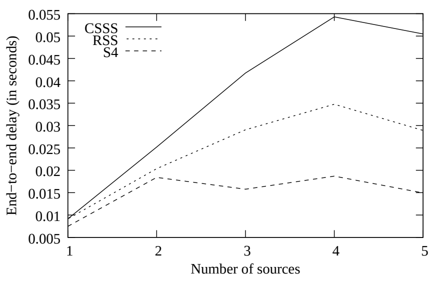

Fig. 6. End-to-end delay as a function of the number of sources, with as many sinks as sources.

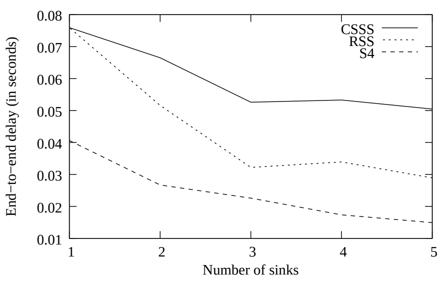

Fig. 7. End-to-end delay with five sources, as a function of the number of sinks.

6 Conclusion

In this paper, we showed that determining distant paths from sources to sinks is an NP-hard problem. Then, we proposed an integer linear program that computes the optimal solutions. We proposed an heuristic called S4 based on realistic assumptions. S4 is a centralized approach that selects sinks and pivots in order to provide distant paths between source-sink pairs and to reduce congestion in the network. Simulation results showed that S4 outperforms the existing strategies in terms of delay (which is reduced by up to 50% in our scenarios) and packet loss (which is reduced by up to 41% in our scenarios). The perspectives of this work include the enhancement of S4 (including the order in which sources are selected). We aim to have a distributed strategy that is able to provide distant paths without requiring the knowledge of the whole topology. Moreover, we plan to simulate S4 for other representative topologies.

Acknowledgment: This work has been partially supported by a research grant from the Lebanese National Council for Scientific Research (LNCSR).

References

1. Kim, H., Seok, Y., Choi, N., Kwon, T.: Optimal multi-sink positioning and energy-efficient routing in wireless sensor networks. In: Information Networking. (2005)
2. Oyman, E.I., Ersoy, C.: Multiple sink network design problem in largescale wireless sensor networks. In: IEEE ICC. (2004)
3. Chang, J., Tassiulas, L.: Maximum lifetime routing in wireless sensor networks. IEEE/ACM Transactions on Networking **12**(4) (2007)
4. Buratti, C., Orris, J., Verdone, R.: On the design of tree-based topologies for mutli-sink wireless sensor nerworks. (September 2006)
5. Kalantari, M., Shayman, M.: Design optimization of multi-sink sensor networks by analogy to electrostatic theory. In: IEEE WCNC. (2006)
6. Hu, W., Bulusu, N., Jha, S.: A communication paradigm for hybrid sensor/actuator networks. Springer International Jounal of Wireless Information Networks **14**(3) (2005)
7. Thepvilojanapong, N., Tobe, Y., Sezaki, K.: Har: Hierarchy-based anycast routing protocol for wireless sensor networks. In: SAINT. (2005)
8. Kim, J., Lin, X., Shroff, N.B.: Minimizing Delay and Maximizing Lifetime for Wireless Sensor Networks With Anycast. In: IEEE INFOCOM. (2008)
9. Marina, M.K., Das, S.R.: On-demand multipath distance vector routing in ad hoc networks. In: ICNP. (2001)
10. Lee, S.J., gerla, M.: Split multipath routing with maximally disjoint paths in ad hoc networks. In: IEEE ICC. (2001)
11. Pearlman, M.R., Haas, Z.J., Sholander, P., Tabrizi, S.S.: On the impact of alternate path routing for load balancing in mobile ad hoc networks. In: ACM Mobile Ad Hoc Networking and Computing. (2000)
12. Saha, D., Toy, S., Bondyopadhyay, S., Ueda, T. anda Tanaka, S.: An adaptive framework for multipath routing via maximally zone-disjoint shortest paths in ad hoc wireless networks with directional antenna. In: Proc. Global Telecommunications. (2003)
13. Wang, Z., Bulut, E., Szymanski, B.K.: Energy Efficient Collision Aware Multipath Routing for Wirelss Sensor Networks. In: IEEE ICC. (2009)
14. El Rachkidy, N., Guitton, A., Misson, M.: Pirat: Pivot Routing for Alarm Transmission in Wireless Sensor Networks. In: IEEE Local Computer Networks. (2009)
15. Qian-Ping, G., Satoshi, O., Shietung, P.: Efficient algorithms for node disjoint path problems. Proceedings of Electronics, Information and Communication Engineers Conference **2** (1994)
16. Perkins, C., Belding-Royer, E., Das, S.: Ad hoc on-demand distance vector (AODV) routing. Request For Comments 3561, IETF (July 2003)