



HAL
open science

Joint sink and route selection for anycast communications in wireless sensor networks

Nancy El Rachkidy, Alexandre Guitton, Michel Misson

► To cite this version:

Nancy El Rachkidy, Alexandre Guitton, Michel Misson. Joint sink and route selection for anycast communications in wireless sensor networks. Introduction to routing issues, 2012. hal-01790291

HAL Id: hal-01790291

<https://hal.science/hal-01790291>

Submitted on 11 May 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Joint sink and route selection for anycast communications in wireless sensor networks

Nancy EL RACHKIDY, Alexandre GUITTON, Michel MISSON

LIMOS CNRS, Clermont Université

Campus des Cézeaux, 63 173 Aubière Cedex, France

Abstract

Most wireless sensor networks are used for monitoring purposes, where some nodes sense the environment and forward their measurements to an entity called the sink. Nodes around the sink tend to have their energy depleted faster than other nodes in the network, as they have the burden of forwarding the measurements from all the nodes. Thus, the current trend in wireless sensor networks is to consider networks with several sinks. In this context, anycast communications are important: each node producing traffic has to send its measurements to any available sink, generally to the closer one.

In this chapter, we aim at reducing the congestion in the whole network by jointly selecting sinks and routes from sources to sinks. We show that considering the two problems separately results into low performance. After proving that the joint sink and route selection problem is NP-hard, we propose a mixed integer linear program in order to obtain optimal solutions. Then, we propose a centralized mechanism that selects sinks and routes packets with reasonable performance (compared to the optimal solutions) and control overhead. Our approach contributes to balance the generated traffic in the network by considering special nodes called pivots. Our mechanism is validated by extensive simulations on generic network topologies. Then, we show how this mechanism can be distributed, using only a limited knowledge.

1 Introduction

In the past few years, wireless sensor networks (WSNs) have been used in several monitoring (He et al., 2004) and tracking (Juang et al., 2002) applications. WSNs are composed of cheap battery-powered devices that are able to sense the environment, to collect the measurements, and to send the collected data in a hop-to-hop 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.

In a typical monitoring application (such as in forest monitoring), a WSN is deployed over a large area. Sensor nodes sense the environment periodically and report their measurements to one sink. High contention is likely to occur around the sink, as it is where the paths from each source converge. In such a scenario, the network might experience high latency and high packet loss rate, which is not acceptable in an emergency scenario. Thus, it is very important to design communication protocols that can reduce the latency and the loss rate. Addition-

ally, high loss rate might yield to a large number of retransmissions, which causes congestion. Minimizing the congestion is therefore an important issue.

As the traffic converges to the sink, nodes close to the sink consume their energy faster than farther nodes. If the nodes close to the sink have their energy depleted, the sink is not able to receive any data and gets disconnected from the network. A solution to this problem is to deploy several sinks. When the traffic is balanced among several sinks, the network lifetime can be significantly increased.

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

In this chapter, we focus on optimizing the selection of the sinks by considering them jointly with the routes. We propose a mechanism which is able to minimize interferences and to reduce the congestion areas between the different paths. This mechanism is first described as centralized (El Rachkidy et al., 2010), and then as distributed. Figure 1 shows an example of a topology with two sources s_1 and s_2 and two destinations d_1 and d_2 , according to three sink selection strategies. On part (a) of the figure, each source is connected to the closest sink, which generates contention around sink d_1 . On part (b), each source is connected to a different sink in order to balance the sink load. However, some nodes of the networks are on the two paths (s_1, d_2) and (s_2, d_1) , which yields to contention on the wireless medium around those nodes. On part (c), the two paths (s_1, d_1) and (s_2, d_2) are distant from each other. The traffic transmitted on one path has little impact on the traffic transmitted on the other path. This third sink selection strategy can only be achieved when considering the sink selection and the routing process simultaneously.

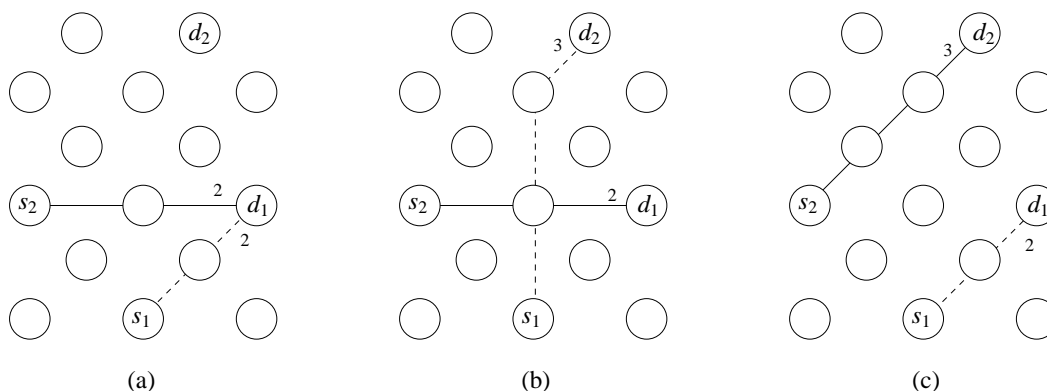


Figure 1: To minimize interferences and reduce congestion on the paths from each source to a sink, sinks and paths have to be considered jointly.

The remainder of this chapter is as follows. In Section 2, we describe the related work and some existing sink selection strategies. Section 3 defines the problem of finding distant paths for a set of anycast communications. We prove that this problem is NP-hard, and we propose an integer linear formulation in order to obtain optimal solutions. Section 4 gives details on the centralized version of our mechanism. Section 5 describes the simulation environment and settings, and provides the simulation results we obtained. Section 6 describes how the mechanism can be distributed. Finally, we conclude our work in Section 7.

2 State of the art

In this section, we focus on four main topics related to this work: the deployment of multiple sinks, the anycast paradigm where sources can send data to any sink, the use of multipaths in wireless routing protocols, and some existing sink selection strategies.

2.1 Multi-sink deployment

Recently, interest is emerging towards deployments with multiple sinks in order to improve the network lifetime and to ensure a fair delivery of data among sinks (Kim et al., 2005; Oyman & Ersoy, 2004), where each sink has identical functional capabilities. Deploying several sinks in the network helps in balancing the traffic among nodes, and the congestion around each sink is reduced: the nodes surrounding the sinks are less overloaded and less used, their energy is less quickly depleted and thus, the network lifetime is extended. Another advantage of deploying multiple sinks is to improve the data gathering by reducing the communication delay from sources to sinks (Chang & Tassiulas, 2007; Buratti et al., 2006) or the total communication cost (Kalantari & Shayman, 2006). Indeed, a network with a single sink might yield to large delays since some nodes are used by most of the paths, and thus several congested links might appear in the network. These congested links cause an increase in the loss rate, and retransmissions become required. A network with several sinks allows to build disjoint paths from sources to sinks, and reduce congestion.

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 WSN can be increased by deploying several sinks, and accessing them using an anycast protocol (Hu et al., 2005; Thepvilojanapong et al., 2005). Indeed, anycast communications can lead to significant energy savings in comparison to traditional protocols (such as Direct Diffusion (Intanagonwiwat et al., 2003)), improve the network scalability, and handle moderate sink mobility. In (Kim et al., 2008), the authors showed that anycast forwarding schemes can substantially reduce the one-hop delay¹ over traditional schemes, especially when nodes are densely deployed, as is the case for many WSN applications. However, the reduction in the one-hop delay may not necessarily lead to a reduction in the expected end-to-end delay experienced by a packet, because the first candidate node that wakes up may not have a small end-to-end delay to the sink.

2.3 Multipath routing

Multipath routing is a feature that enables a source to send packets to a destination through multiple 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 or delayed due to a node failure for instance, the destination is still able to receive packets from the source as long as at least one path is active. Using multipath also helps balancing the energy consumption among the nodes of the network, and therefore extends network lifetime. Most of the multipath routing protocols are based on classic on-demand single path routing methods (Marina & Das, 2001; Lee & Gerla, 2001).

Disjoint multipath routing methods try to determine disjoint paths, *i.e.*, paths that do not have nodes or links in common. As stated in (Pearlman et al., 2000), using disjoint multipaths does not remove the potential for collisions, and may therefore result into large packet loss rates and reduced data transmission performance. The main reason is that wireless transmissions might interfere communications between distant nodes.

¹The one-hop delay is defined as the time required to transmit data to a neighbor. If neighbors have their own sleeping schedule, a sender might have to wait for a neighbor to wake up before sending it a packet.

In (Saha et al., 2003), the authors aim to find zone-disjoint multipaths using directional antennas. A promising approach is described in (Wang et al., 2009), 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 discovered route.

In this chapter, 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 Section 3, the problems of finding two disjoint multipaths between one source-sink pair, or two disjoint paths between two source-sink pairs, are closely related.

In (El Rachkidy et al., 2009), we proposed a probabilistic proactive routing protocol based on pivots and called PiRAT. 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 might become congested and several alarm packets might be dropped. By selecting randomly distant pivot nodes for each source, PiRAT is able to reduce the congested areas and to improve the network performance in terms of delay and packet loss. This use of multipaths allows diversity in routing and avoids congested areas in the network, which contributes to balance the traffic load and the energy consumption between nodes. Figure 2 shows the usage of links in PiRAT. The sources are the nine nodes at the bottom left-hand corner, and the destination is at the top right-hand corner. The pivot nodes selected by the sources are colored in gray. It can be noticed that routes from the sources to the sinks avoid the central area in order to reduce congestion (the shortest path uses the central area). The probabilistic nature of PiRAT can be seen as each node uses several paths to reach a given destination (the destination being either the pivot node or the sink). Although several nodes are involved in the routing process, the amount of transmitted packet per node is small. This leads to reduce the overloaded paths and balance the energy consumption of the nodes. This phenomenon greatly improves the network lifetime. However, PiRAT is not able to reduce the congestion around the sink.

2.4 Existing sink selection strategies

In the random sink selection strategy (RSSS), sinks are randomly chosen (Shukla & Menghanathan, 2005) by the sources. Packets are routed using the shortest path from each source to its selected sink. Generally, this strategy does not perform well in terms of delay and packet loss, as congestion might occur in several areas of the network. Congestion also increases the delay due to retransmission attempts.

Figure 3 shows an example of RSSS on a grid network with three sources (s_1, s_2, s_3) and three sinks (d_1, d_2, d_3) . For each source, a random sink is associated. It can be noticed that the three paths are not distant, and some paths even intersect, which yields to a large congested area in the network. This congested area negatively impacts the network performance in terms of loss rate and end-to-end delay.

In the closest sink selection strategy (CSSS), each source is connected to its closest sink (Luo et al., 2008), in terms of geographical distance or, more generally, hop count. With this strategy, several sources can have the same sink. Thus, the area around such sinks might be congested, leading nodes in these areas to deplete their energy faster than the other nodes in the network, and also reducing the performance due to congestion.

Figure 4 shows an example of CSSS on the same graph. With CSSS, each source chooses its closest sink. In this example, all the sources choose as destination sink d_1 . This strategy yields to a congestion area around d_1 . Note that when an event occurs, it is frequent that several sources located in a limited region start producing packets, and thus choose the same sink.

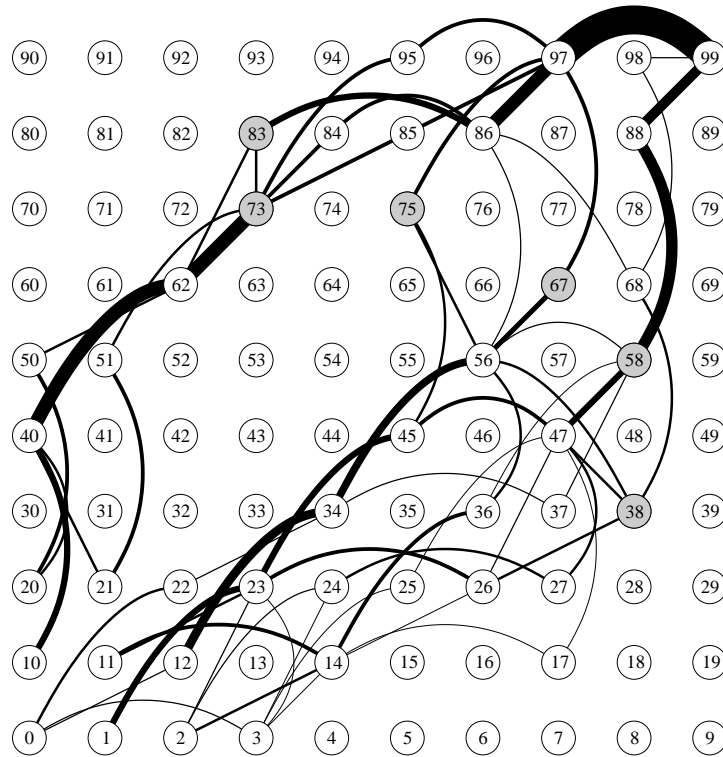


Figure 2: Link usage with the PiRAT protocol (El Rachkidy et al., 2009).

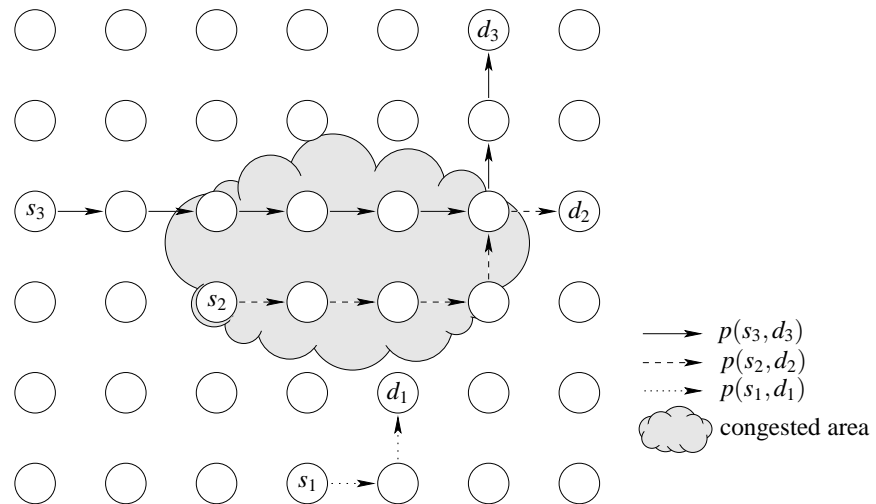


Figure 3: RSSS might cause congestion as paths are not necessarily disjoint or distant.

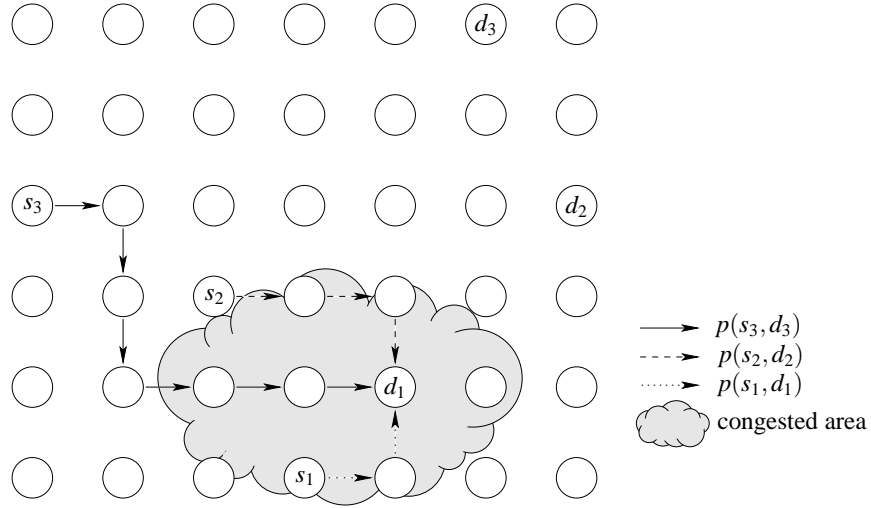


Figure 4: CSSS might cause congestion when several sources are close to the same sink.

3 Mathematical model

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 directly 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. Let us denote by $h(x, y)$ the hop count, and by $d(x, y)$ the distance, between x and y . The minimum hop count between two paths p_1 and p_2 , denoted by $h(p_1, p_2)$, can be defined as:

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

A similar definition can be given for the minimum distance $d(p_1, p_2)$ between two paths p_1 and p_2 .

Definition 1 *The sink selection problem for anycast wireless communications consists in finding for each source $s_i \in S$ a destination $d_{f(i)} \in D$.*

Definition 2 *The routing problem for anycast wireless communications consists in finding, for each pair (s_i, d_i) , where $s_i \in S$ and $d_i \in D$, a path p_i that connects s_i to d_i .*

Definition 3 *The sink selection and routing problem for anycast wireless communications (SSRPAW) consists in associating a destination $d_{f(i)}$ to each source s_i , and in finding a set of paths $\{p_i\}$ that connects each source $s_i \in S$ to its 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. In this definition, 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. A more accurate definition would use minimum distance between paths rather than minimum hops between paths, but the information on the actual distance between paths is usually difficult to obtain.

In the remainder of this section, we study the decision problem associated to SSRPAW, *i.e.*, the problem of determining if there exists a set of paths such that the minimum hop count between the paths of this set is greater than δ . We show that this decision problem is NP-complete. Then, we propose an integer linear program that allows to find paths that are distant.

3.1 Proof of the NP-completeness

In order to prove that SSRPAW is NP-complete, let us first define a similar problem.

Definition 4 *The set-to-set disjoint path problem takes as input a graph $G = (V, E)$, a set S of k sources and a set D of k destinations. It consists in determining if there are k mutually node-disjoint paths $\{p_i\}$, such that each 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 (Qian-Ping et al., 1994). It is similar to the node-to-node disjoint path problem.*

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

Proof 1 *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 \tilde{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 $\tilde{G} = (\tilde{V}, \tilde{E})$ is the following. Each node $n \in V$ is also a node of \tilde{V} . Each edge $e = (x, y) \in E$ becomes a path of δ edges in \tilde{E} , connecting $x \in \tilde{V}$ to $y \in \tilde{V}$.

Let us now assume that SSRPAW can be solved in polynomial time in this graph \tilde{G} . This means that there are $|S| = k$ paths $\{\tilde{p}_i\}$ in \tilde{G} , for $1 \leq i \leq k$, such that each path \tilde{p}_i connects a source $s_i \in S$ to a sink $d_{j_i} \in D$. Moreover, for any $i_1 \neq i_2$, $h_{\tilde{G}}(\tilde{p}_{i_1}, \tilde{p}_{i_2}) \geq \delta$, by definition of SSRPAW. By construction of \tilde{G} , each path \tilde{p} in \tilde{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 $h_G(p_{i_1}, p_{i_2}) = h_{\tilde{G}}(\tilde{p}_{i_1}, \tilde{p}_{i_2})/\delta \geq \delta/\delta = 1$, 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 ILP formulation

The goal of this subsection is to compute optimal solutions by an integer linear program. This program takes as input the following parameters: a set of nodes V , a set of sources $S \subset V$, a set of sinks $D \subset V$, a set of binary variables $e_{x,y}$ representing the edges, and a set of variables $dist(x, y)$ representing the distances (in hop count or in meters) between nodes. 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_s in D , such that the paths are distant from each other and that the overall length of paths is small. Each path is defined as a set of binary variables $p_s(x, y)$, such that $p_s(x, y)$ is equal to 1 if path p_s uses edge (x, y) , and 0 otherwise. The objective function and the ILP are given on Table 1.

The objective function uses a parameter γ to combine the minimization of the total length of the paths with the maximization of the average distance between the paths (computed using the variable called *gap*).

Equation (1) states that each path has to use edges that are in the graph. Equation (2) forces every source s to be the start of path p_s . Similarly, Equation (3) forces every path p_s to end in a destination $d \in D$. Equation (4)

$$\text{minimize } \gamma \left(\sum_{s \in \mathcal{S}} \sum_{x \in V} \sum_{y \in V} p_s(x, y) \right) + (1 - \gamma) \left(- \sum_{s_1 \in \mathcal{S}} \sum_{s_2 \in \mathcal{S} \setminus \{s_1\}} \text{gap}(s_1, s_2) \right)$$

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

$$\forall s \in \mathcal{S} \quad \sum_{y \in V} (p_s(s, y) - p_s(y, s)) = 1 \quad (2)$$

$$\forall s \in \mathcal{S} \quad \sum_{x \in D} \sum_{y \in V} (p_s(x, y) - p_s(y, x)) = -1 \quad (3)$$

$$\forall s \in \mathcal{S}, x \in V \setminus D \setminus \{s\} \quad \sum_{y \in V} (p_s(x, y) - p_s(y, x)) = 0 \quad (4)$$

$$\forall s_1 \in \mathcal{S}, s_2 \in \mathcal{S} \setminus \{s_1\}, \forall x \in V, \forall y \in V \quad \text{gap}(s_1, s_2) \leq \text{dist}(x, y) + \#D \cdot (1 - \sum_{x_p \in V} p_{s_1}(x, x_p)) + \#D \cdot (1 - \sum_{y_p \in V} p_{s_2}(y, y_p)) \quad (5)$$

$$\forall s_1 \in \mathcal{S}, s_2 \in \mathcal{S} \setminus \{s_1\}, \forall x \in V, \forall y \in V \quad \text{gap}(s_1, s_2) \leq \text{dist}(x, y) + \#D \cdot (1 - \sum_{x_p \in V} p_{s_1}(x_p, x)) + \#D \cdot (1 - \sum_{y_p \in V} p_{s_2}(y_p, y)) \quad (6)$$

$$\forall s_1 \in \mathcal{S}, s_2 \in \mathcal{S} \setminus \{s_1\} \quad \text{gap}(s_1, s_2) \geq \sum_{x \in V} \sum_{y \in V} (\text{dist}(x, y) \cdot \text{prod}_{s_1, s_2}(x, y)) \quad (7)$$

$$\forall s_1 \in \mathcal{S}, s_2 \in \mathcal{S} \setminus \{s_1\} \quad \sum_{x \in V} \alpha_{s_1, s_2}(x) = 1 \quad (8)$$

$$\forall s_1 \in \mathcal{S}, s_2 \in \mathcal{S} \setminus \{s_1\}, x \in V \quad \alpha_{s_1, s_2}(x) \leq \sum_{y \in V} \text{path}_{s_1}(x, y) \quad (9)$$

$$\forall s_1 \in \mathcal{S}, s_2 \in \mathcal{S} \setminus \{s_1\} \quad \sum_{y \in V} \beta_{s_1, s_2}(y) = 1 \quad (10)$$

$$\forall s_1 \in \mathcal{S}, s_2 \in \mathcal{S} \setminus \{s_1\}, y \in V \quad \beta_{s_1, s_2}(y) \leq \sum_{z \in V} \text{path}_{s_2}(y, z) \quad (11)$$

$$\forall s_1 \in \mathcal{S}, s_2 \in \mathcal{S} \setminus \{s_1\}, \forall x \in V, \forall y \in V \quad \text{prod}_{s_1, s_2}(x, y) \leq \alpha_{s_1, s_2}(x) \quad (12)$$

$$\forall s_1 \in \mathcal{S}, s_2 \in \mathcal{S} \setminus \{s_1\}, \forall x \in V, \forall y \in V \quad \text{prod}_{s_1, s_2}(x, y) \leq \beta_{s_1, s_2}(y) \quad (13)$$

$$\forall s_1 \in \mathcal{S}, s_2 \in \mathcal{S} \setminus \{s_1\}, \forall x \in V, \forall y \in V \quad 1 - \alpha_{s_1, s_2}(x) - \beta_{s_1, s_2}(y) + \text{prod}_{s_1, s_2}(x, y) \geq 0 \quad (14)$$

Table 1: Integer linear constraints.

ensures that for every node x in a path p_s , the number of edges arriving in x on p_s is equal to the number of edges leaving x on p_s (apart from the source s and the destination of p_s).

Variable $gap(s_1, s_2)$ denotes the distance between the two paths p_{s_1} and p_{s_2} , which is defined as the minimum distance between two nodes $x \in p_{s_1}$ and $y \in p_{s_2}$. Equations (5) and (6) state that $gap(s_1, s_2)$ is lower than or equal to the distance between two nodes $x \in p_{s_1}$ and $y \in p_{s_2}$, where $\#D$ is the diameter of the graph. Note that if $x \notin p_{s_1}$ (or if $y \notin p_{s_2}$), the right-hand part of the equation becomes larger than $\#D$, and the variable $gap(s_1, s_2)$ is thus unconstrained for this x (or y). Equations (7) to (14) state that there exists a pair (x, y) such that $x \in p_{s_1}$ and $y \in p_{s_2}$, and such that $gap(s_1, s_2) \geq dist(x, y)$. Since $gap(s_1, s_2)$ is lower than or equal to any distance between the two paths, and since $gap(s_1, s_2)$ is greater than or equal to one distance between the two paths, $gap(s_1, s_2)$ is the minimum distance between the two paths. The fact that the pair (x, y) exists is determined by the fact that α_{s_1, s_2} is equal to one for a single x , and that β_{s_1, s_2} is equal to one for a single y . Indeed, for every pair (s_1, s_2) , there is only one x and one y such that $\alpha_{s_1, s_2}(x) = 1$ and $\beta_{s_1, s_2}(y) = 1$, according to Equations (8) and (10). Equations (9) and (11) ensure that $x \in p_{s_1}$ and that $y \in p_{s_2}$. Finally, Equation (7) states that $gap(s_1, s_2) \geq dist(x, y)$ for this x and for this y , that is $gap(s_1, s_2) \geq \sum_{x \in V} \sum_{y \in V} (dist(x, y) \cdot \alpha_{s_1, s_2}(x) \cdot \beta_{s_1, s_2}(y))$. Equations (12) to (14) allow us to write the non-linear product $prod_{s_1, s_2}(x, y) = \alpha_{s_1, s_2}(x) \cdot \beta_{s_1, s_2}(y)$ as a set of linear equations. When combining Equations (7) to (14) with Equations (5) and (6), $gap(s_1, s_2)$ contains the minimum distance between the two paths p_{s_1} and p_{s_2} .

4 Centralized mechanism for joint sink and route selection

This section describes in details the mechanism we propose in order to reduce the congestion in the network, and especially around the sinks. Our mechanism is called S4, for Simultaneous Sink Selection Strategy (El Rachkidy et al., 2010). S4 is a centralized approach combining a sink selection strategy with a routing mechanism. S4 uses a greedy algorithm in order to select sinks and to compute paths: each source is considered sequentially. For each source, a sink and an intermediate node, called the pivot, are selected. Packets are forwarded from the source to the pivot first, and from the pivot to the sink. The goal is to make paths as disjoint as possible with few modifications to the routing protocol.

The sink and pivot selection is 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 computes the path $p(s, x, d)$ from s to d via x . S4 chooses for source s the pivot x' that minimizes the number of nodes in common between $p(s, x', d)$ and \mathcal{P} . If there are several such paths, S4 chooses the one that minimizes the number of hops in $p(s, x', d)$. Finally, if there are still several such paths, S4 chooses one of them arbitrarily. The aim of S4 is to reduce the energy consumption since it balances the traffic in the whole network.

In order to operate S4, we consider in this section that there is a centralized entity which knows the whole topology and computes paths for all the sources. For each source s , this entity has to compute the shortest paths from s to any potential pivot (that is, to any node), and the shortest paths from any pivot to any sink. The first computation requires $O(n + m)$ operations, where n is the number of nodes and m is the number of edges, and the second requires $O(n + m)$ too (note that a single computation is performed for all the sinks by assigning an initial weight of 0 to all sinks). Then, all the nodes are considered as pivots and all the sinks are considered as destinations. Thus, the overall complexity is $O(|S| \cdot (n + m) + |S| \cdot |D| \cdot n^2)$.

Figure 5 shows an example of S4. We consider the same example as for RSSS and CSSS (see Figure 3 and Figure 4). Each source computes the distance to each sink by using pivot nodes as relay. s_1 chooses d_1 as sink and S4 stores the path $p(s_1, d_1)$ in \mathcal{P} . Then, the second potential source s_2 chooses to send its data to d_2 . s_2 builds the path $p(s_2, d_2)$ based on \mathcal{P} . s_2 tries to avoid the nodes that are already in use by s_1 and verifies if the path to

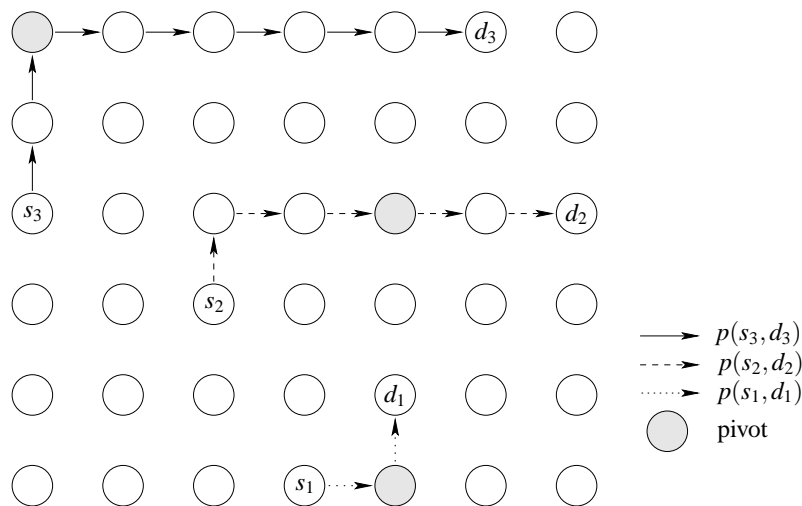


Figure 5: S4 avoids congested zones by using pivots.

d_2 is distant enough from $p(s_1, d_1)$. S4 adds the nodes used by $p(s_2, d_2)$ to \mathcal{P} . Then, s_3 tries to find a sink that is not in use yet and checks if it can send data to this sink by using nodes that are not stored in \mathcal{P} and that are distant enough from the other existing paths. S4 selects pivots yielding to paths that are distant enough in order to reduce interference between paths and congestion in the whole network. Paths between sources and sinks are not necessarily shortest paths, but they are distant (when it is possible).

5 Simulations

This section is divided into two parts. First, we compare the distance between paths computed by S4 with the distance between paths computed using our ILP and by RSSS and CSSS. This comparison is applied on small network topologies of 20 nodes (in order for the ILP to find optimal solutions in a reasonable time). Second, we compare the network performance of S4 with those of RSSS and CSSS on larger topologies.

5.1 Simulations on small topologies

Here, we compare S4 with three strategies: the ILP, RSSS and CSSS. Due to the time required by ILP to find optimal solutions, we only consider small topology of 20 nodes randomly distributed on a 100×100 square meters topology. The communication range is set to 30 m. Results are averaged over 100 repetitions. γ is set to one eleventh: the gap between paths (in terms of distance) is considered to be ten times more important than the total distance of the paths.

The metric used for comparison is the average distance between paths, defined in the following way. Each strategy produces a set of paths \mathcal{P} . For each path p of the set, the distance $d_{\mathcal{P}}(p)$ to the closest path p' of the set is computed. The distance between two paths p_1 and p_2 is equal to the minimum distance between any nodes $x \in p_1$ and $y \in p_2$. Then, the average distance between paths is the average of the distances of all the paths, $\sum_{p \in \mathcal{P}} d_{\mathcal{P}}(p) / \text{card}(\mathcal{P})$, where $\text{card}(\mathcal{P})$ is the number of paths in set \mathcal{P} . Note that if two paths p_1 and p_2 have a node in common, $d(p_1, p_2) = 0$ and thus, $d_{\mathcal{P}}(p_1) = 0$.

Figure 6 shows the average distance between paths as a function of the number of sources, with as many

sinks as sources, for the four strategies. The distance is measured in meters. We notice that the maximum average distance is reached when there are two sources and two destinations. In this case, the average distance is 8 meters for ILP, 6 meters for S4, 2 meters for CSSS, and 1.8 meters for RSSS. For small network topologies, when the number of sources and sinks increases, the distance between paths decreases as the paths cannot be distant. We also notice that paths in S4 are only 25% less distant than in ILP. RSSS and CSSS both compute paths that are close to each other, and thus that yield congestion.

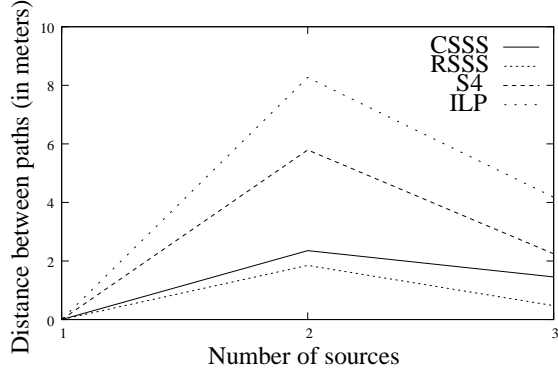


Figure 6: Average distance between paths, with as many sinks as sources.

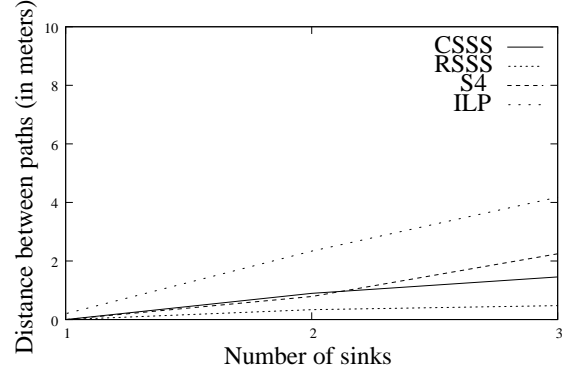


Figure 7: Average distance between paths, with three sources.

Figure 7 shows the average distance between paths as a function of the number of sinks, for the four strategies. The number of sources is constant and set to 3. We notice that the average distance increases with the number of sinks. This is due to the fact that by increasing the number of sinks, the possibilities of having distant paths increase. As expected, the ILP formulation shows the better performance. S4 is able to reach 75% of the performance of ILP. The maximum average distance is obtained when the number of sinks equals to the number of sources.

5.2 Simulations on large topologies

In this section, we describe the simulations we ran in order to compare S4 with RSSS and CSSS on larger topologies. We use version 2.31 of the NS-2 simulator. We use IEEE 802.15.4 physical and MAC layers with the non-beacon enabled mode. The propagation model is the probabilistic two ray ground model, with default parameters. The transmission power is set to a realistic value of -25 dBm, which yields a communication range of about 25 m. The size of the nodes queue is set to 50 packets.

In the simulations, we considered topologies of 100 nodes, randomly distributed on an area of 100×100 meter square. All sensors are full function devices with routing capabilities. The PAN coordinator is located at the center of the area. We wait for the network to be fully associated before injecting data packets. Data packets of 77 bytes (at the physical layer) are generated during 50 seconds, at a rate varying between 0.5, 1, and 2 packets per source and per second. The routing protocol used for the three strategies is AODV² (Perkins et al., 2003). Notice that before AODV can send packets to an unknown destination, it has to establish a path through reply and request messages, which introduces delay³. In order to have consistent results (especially for RSSS), results are averaged over 100 repetitions.

²AODV is slightly modified in S4 in order to allow pivot nodes

³For RSSS 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.

5.2.1 Performance metrics

Our mechanism is evaluated and compared to RSSS and CSSS according to the following performance metrics: average distance (computed as in 5.1), packet loss (computed as the number of received packets over the number of generated packets), and end-to-end delay (defined as the time interval between a packet transmission and its reception by the sink). In the following, figures show the metric performances by considering 100 nodes randomly spread in the network. Similar results are obtained by considering grid networks of 49 nodes (El Rachkidy et al., 2010).

5.2.2 Average distance between paths

Figure 8 shows the average distance between paths for the three strategies, as a function of the number of sources, with a number of sinks equal to the number of sources. The average distance reaches a maximum for two sources: as the number of sources increases, more paths have to be computed, which decreases the average distance between paths. RSSS computes paths that often intersect, which rapidly decreases the average distance. S4 is able to build paths whose average distance is between 13 and 20 meters (with our settings), which limits interferences among paths, and thus improves overall performance.

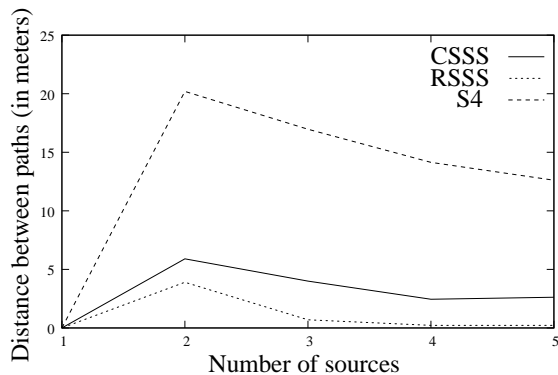


Figure 8: Average distance between paths, with as many sinks as sources.

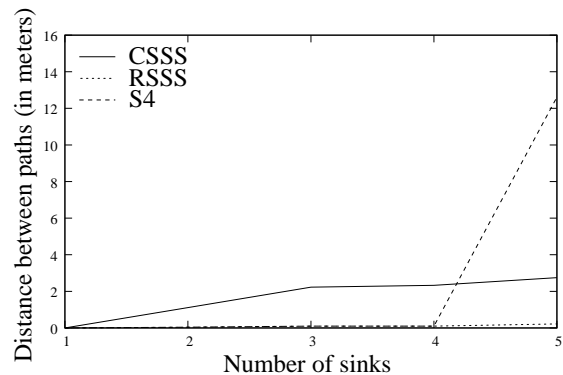


Figure 9: Average distance between paths, with five sources.

Figure 9 shows the average distance between paths for the three strategies, as a function of the number of sinks, with five sources. None of the strategies are able to perform very well with these settings, as the number of sinks is smaller than the number of sources. The fact that CSSS is able to produce paths that have an average distance greater than zero means that, on average, one source (out of the five) was assigned to its own sink. While the distance for the four other paths might be zero, the distance between the fifth path (with its own sink) and the other paths is not zero, which makes the average distance not zero as well. S4 is only able to produce distant paths when the number of sinks is high enough, and in our case, when it is equal to the number of sources.

5.2.3 Packet loss

Figure 10 and Figure 11 show the average packet loss as a function of the number of sources and sinks, with a data rate equals to 2 packets per source and per second. 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. Also, paths cannot be as distant as when the number of sources is small. S4 is able to significantly reduce the

packet loss compared to RSSS and CSSS, especially when there are many sinks. 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. When the number of sources and sinks is small, the use of pivots increases the length of paths, and S4 exhibits worse performance than RSSS and CSSS. However, for five sources and five sinks, S4 reduces by approximately 25% the packet loss probability of RSSS and 50% the packet loss probability of CSSS. S4 also outperforms the other two strategies when there are five sources and one sink: S4 reduces by approximately 46% the packet loss probability of RSSS and CSSS for five sources and one sink.

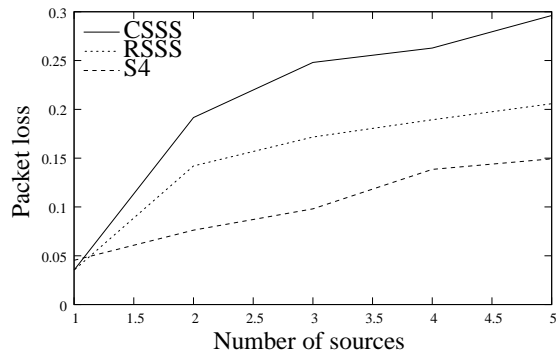


Figure 10: Packet loss rate, with as many sinks as sources.

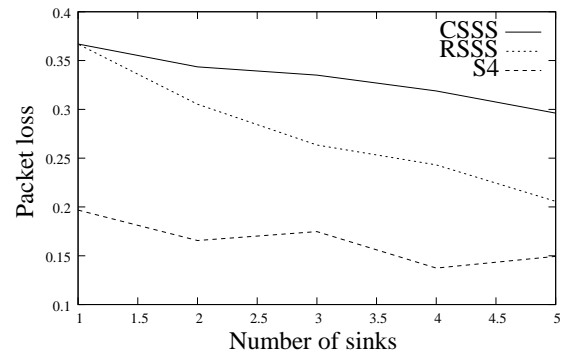


Figure 11: Packet loss rate, with five sources.

5.2.4 End-to-end delay

Figure 12 shows the average end-to-end delay for the three strategies, as a function of the number of sources, with a number of sinks equal to the number of sources and with a data rate equals to 2 packets per source and per second. The number of nodes used in the network is 100. For RSSS, 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). Surprisingly, it can be noticed that CSSS induces larger delays than RSSS. This is explained by the fact that CSSS tends to select the same sink for several close sources, which yields to congested areas around these sinks. RSSS balances the sink usage by choosing sinks randomly. While CSSS and RSSS have almost the same packet loss (see Figure 10 and Figure 11), the impact on delay is significant: packets with large delay are more likely to be dropped in RSSS, 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 the sink selection and the route establishment jointly. S4 reduces the average end-to-end delay over CSSS by 66% and over RSSS by 48%, for five sources and five sinks.

Figure 13 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 Figure 11). S4 outperforms the two other strategies, even with one sink: in this case, it reduces the end-to-end delay of CSSS and RSSS by 70%. This is achieved by choosing pivots that force the five paths to be distant (except at the sink where they meet), and thus reducing interferences among these paths.

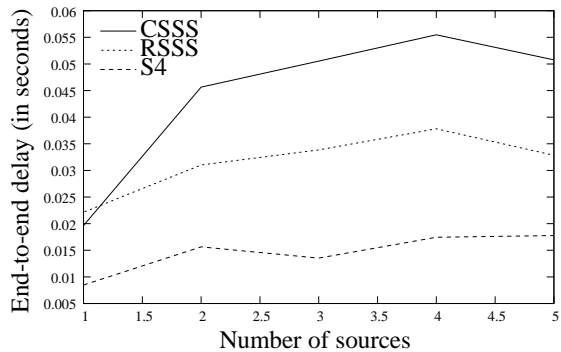


Figure 12: End-to-end delay, with as many sinks as sources.

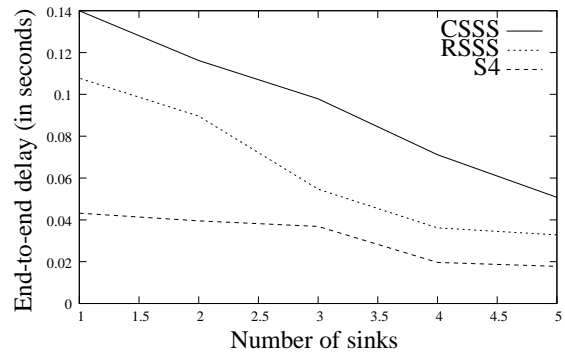


Figure 13: End-to-end delay, with five sources.

5.3 Network performance with a varying data rate

In this subsection, we study the impact of the data rate on the network. We always consider a network of 100 nodes randomly deployed. We set the number of source as well as the number of sinks to three. Then, we study the behavior of the three strategies (CSSS, RSSS, and S4) in the network.

Figure 14 shows the average packet loss in terms of the data rate. The packet loss increases by increasing the traffic load in the network. We notice that S4 outperforms CSSS by 60% and RSSS by 33% for a data transmission rate of 2 packets per source and per second. We also notice that even with small data rate (0.5 packet per source and per second), S4 has the best behavior and shows a gain of 65% compared to CSSS and 50% compared to RSSS. Figure 15 shows the average end-to-end delay as a function of the data rate for the three strategies. As the data rate increases, the end-to-end delay decrease because the packet loss becomes higher and several packets are dropped. The packets most likely to be dropped are those that correspond to long routes. We also notice that S4 shows a gain of 74% (respectively 61%) compared to CSSS and 60% (respectively 59%) compared to RSSS for a data rate of 2 (respectively for a data rate equals to 0.5).

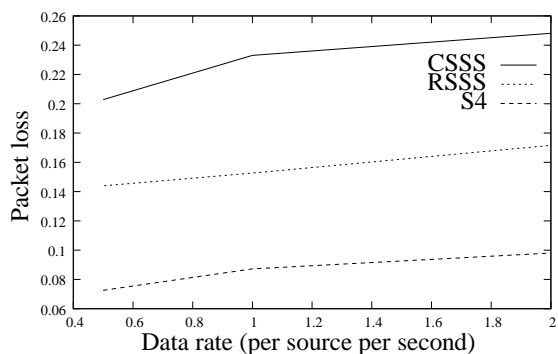


Figure 14: End-to-end delay, with three sources and three sinks.

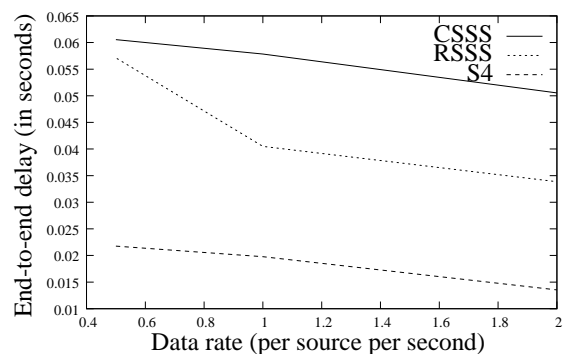


Figure 15: End-to-end delay, with three sources and three sinks.

6 Distributed version of S4

In the centralized version of S4, we made the following assumptions. First, the entity that performs the computation knows the whole topology, in order to compute the paths from any source to any destination via any pivot, or the hop counts. Second, sources were considered sequentially by a central entity. Third, the topology is static and nodes do not fail.

In the following, we describe how the centralized assumptions can be removed in order to make S4 a distributed protocol. The main requirements of the distributed version of S4 are the following. First, a local computation is performed by every source. Second, each source knows the set of all sinks. Note that no node is required to know the whole topology.

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 ZigBee (ZigBee, 2008) for example). Similarly, the number of nodes in common between two paths can be computed by having s send a first message to x , and a second message via x to d . Each node on both paths can send a notification back to s , containing the path traveled. With this information, s 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 and that s has a knowledge of the network parameters (such as maximum number of child routers per node, maximum number of children per node, and maximum depth). We assume that hierarchical addresses are used, which avoids the use of additional control messages.

Sources advertise themselves to other neighbor sources, using a limited flooding protocol. Ignoring sources that are far away might be suboptimal when considering routes and sinks, but generally, it is better to avoid very long paths, even if they are distant to other paths. Once sources know their neighbor sources, they schedule their pivot and sink selection according to any order (which can depend on their address, for instance). Once a source has computed its own pivot and sink, it broadcasts this information to all neighbor sources. Then, the next sources in the order compute their own pivots and sinks, until all the sources have performed this task.

The determination of pivots by a source is achieved by using a limited flooding (common to each potential sink). The flooding is restricted to nodes that are close to the shortest path from the source to one of the sink. Indeed, it is not efficient to consider pivots that are too far away from the shortest path from the source to one sink.

To summarize, each source sends the three following messages in order to compute its pivot and sink. First, the source identifies neighbor sources by using a flooding limited to the region around the source. Second, the source identifies potential pivots by using a flooding limited to the nodes that are close to the shortest path from the source to one of the sink. Third, the source informs the neighbor sources of its selected pivot and sink by using a flooding, limited to the region around the source.

When the topology changes, the routing protocol upon which S4 is built detects the change and modifies the routes. Such changes can cause initially distant routes to become closer or even to overlap. Such overlaps can reduce the performance of S4. To avoid this problem, it is important for each source to recompute periodically the set of potential pivots. However, a trade-off has to be made between the control overhead caused by determining pivots, and the performance reduction caused by keeping inefficient pivots in case of topology changes.

Additionally, the failure of pivots has a negative impact on S4, as packets are routed to pivots before being routed to the sink. A pivot failure has to be reported to the source by the routing protocol (using notifications indicating a failure to deliver to the pivot) or by the sink (which stops receiving messages from the source). Upon receiving this report, the source can choose another pivot from the set of previously computed potential pivot. If this set is empty, the source has to initiate another pivot detection.

7 Conclusion

In this chapter, we showed that it is important to perform jointly sink selection and route computation in an anycast, multi-sink deployment. We showed that determining distant paths from sources to sinks is an NP-hard problem, and we proposed an integer linear program that computes optimal solutions, using the global knowledge of the topology and heavy processing power. Then, we proposed an heuristic called S4 based on realistic assumptions. S4 is an approach that selects sinks and pivots to provide distant paths between source-sink pairs, in order 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). S4 is also shown to produce paths that are on average distant from each other. We conclude this work by proposing a method to distribute S4, which requires a limited amount of control messages and a partial vision of the topology.

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

References

- Buratti, C., Orris, J., & Verdone, R. (2006). On the design of tree-based topologies for multi-sink wireless sensor networks.
- Chang, J. & Tassiulas, L. (2007). Maximum lifetime routing in wireless sensor networks. *IEEE/ACM Transactions on Networking*, 12(4).
- El Rachkidy, N., Guitton, A., & Misson, M. (2009). Pirat: Pivot routing for alarm transmission in wireless sensor networks. In *IEEE Local Computer Networks*.
- El Rachkidy, N., Guitton, A., & Misson, M. (2010). Routing protocol for anycast communications in a wireless sensor network. In *IFIP Networking*, LNCS.
- He, T., Krishnamurthy, S., Stankovic, J. A., Abdelzaher, T., Luo, L., Stoleru, R., Yan, T., Gu, L., Hui, J., & Krogh, B. (2004). Energy-efficient surveillance system using wireless sensor networks. In *International Conference on Mobile Systems, Applications and Services* (pp. 270–283).
- Hu, W., Bulusu, N., & Jha, S. (2005). A communication paradigm for hybrid sensor/actuator networks. *Springer International Journal of Wireless Information Networks*, 14(3).
- Intanagonwiwat, C., Govindan, R., Estrin, D., Heidemann, J., & Silva, F. (2003). Direct diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking (TON)*, 11, 2–16.
- Juang, P., Oki, H., Wang, Y., Martonosi, M., Pen, L. S., & Rubenstein, D. (2002). Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet. *ACM SIGOPS Operating Systems Review*, 36(5), 96–107.
- Kalantari, M. & Shayman, M. (2006). Design optimization of multi-sink sensor networks by analogy to electrostatic theory. In *IEEE WCNC*.
- Kim, H., Seok, Y., Choi, N., & Kwon, T. (2005). Optimal multi-sink positioning and energy-efficient routing in wireless sensor networks. In *Information Networking*.

- Kim, J., Lin, X., & Shroff, N. B. (2008). Minimizing delay and maximizing lifetime for wireless sensor networks with anycast. In *IEEE INFOCOM*.
- Lee, S. J. & Gerla, M. (2001). Split multipath routing with maximally disjoint paths in ad hoc networks. In *IEEE ICC*.
- Luo, D., Zuo, D., & Yang, X. (2008). An optimal sink selection scheme for multi-sink wireless sensor networks. In *Computer Science and Information Technology ICCSIT*.
- Marina, M. K. & Das, S. R. (2001). On-demand multipath distance vector routing in ad hoc networks. In *ICNP*.
- Oyman, E. I. & Ersoy, C. (2004). Multiple sink network design problem in largescale wireless sensor networks. In *IEEE ICC*.
- Pearlman, M. R., Haas, Z. J., Sholander, P., & Tabrizi, S. S. (2000). On the impact of alternate path routing for load balancing in mobile ad hoc networks. In *ACM Mobile Ad Hoc Networking and Computing*.
- Perkins, C., Belding-Royer, E., & Das, S. (2003). *Ad hoc On-Demand Distance Vector (AODV) Routing*. Request For Comments 3561, IETF.
- Qian-Ping, G., Satoshi, O., & Shietung, P. (1994). Efficient algorithms for node disjoint path problems. *Proceedings of Electronics, Information and Communication Engineers Conference, 2*.
- Saha, D., Toy, S., Bondyopadhyay, S., Ueda, T., & Tanaka, S. (2003). An adaptive framework for multipath routing via maximally zone-disjoint shortest paths in ad hoc wireless networks with directional antenna. In *Proc. Global Telecommunications*.
- Shukla, I. & Menghanathan, N. (2005). Impact of leader selection strategies on the PEGASIS data gathering protocol for wireless sensor networks. *Ubiquitous Computing and Communication Journal*.
- Thepvilojanapong, N., Tobe, Y., & Sezaki, K. (2005). Har: Hierarchy-based anycast routing protocol for wireless sensor networks. In *SAINT*.
- Wang, Z., Bulut, E., & Szymanski, B. K. (2009). Energy efficient collision aware multipath routing for wireless sensor networks. In *IEEE ICC*.
- ZigBee (2008). *ZigBee Specification*. Standard ZigBee 053474r17, ZigBee Standards Organization.