Computer Networks 207 (2022) 108832

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

SplitPath: High throughput using multipath routing in dual-radio Wireless Sensor Networks

Nildo dos Santos Ribeiro Junior [a], Marcos A.M. Vieira [a], Luiz F.M. Vieira [a,*], Omprakash Gnawali [b]

[a] Department of Computer Science, Universidade Federal de Minas Gerais, 31270-90, Belo Horizonte, Minas Gerais, Brazil
[b] Department of Computer Science, University of Houston, 77204-3010, Houston, TX, USA

ARTICLE INFO

Keywords:
Multipath routing
Wireless Sensor Networks
Dual-radio

ABSTRACT

Dual-Radio platforms were proposed to improve the throughput of Wireless Sensor Network applications while conserving energy efficiency. However, current dual-radio protocols do not use all the hardware available. We model this problem as the minimum disjoint parity paths problem. We present the design, implementation and evaluation of SplitPath, a distributed routing protocol that computes two vertex-disjoint paths with the same parity. Unlike previous work on multipath routing, SplitPath is the first protocol to use multiple paths in dual-radio WSNs to achieve maximum throughput. The protocol was evaluated with experiments in the physical world. We compare our proposal with FastForward, the state-of-the-art protocol for dual-radio. Our approach improved the throughput by 60%, and achieved 96% of the maximum theoretical limit.

1. Introduction

Wireless Sensor Networks (WSNs) consist of tiny sensor nodes, which act as both data generators and network relays. Each node has one or more sensors, a microprocessor and radio transceivers [1]. Wireless communication provides ease of deployment, and the distributed sensing capabilities makes WSNs an important component of our daily lives [2]. They have many applications, such as environmental monitoring, agriculture, health care, smart buildings [3], and Social IoT [4]. In each scenario, the environment, the application's design principles, the hardware and the system's constraints might be very different [5]. For traditional applications, three key issues have been taken into account in their design: cost, memory usage and total power consumption.

Minimizing power consumption at the expense of the network's performance is a well known tradeoff in the design of WSNs. The design of traditional sensor network platforms has favored low power operation at the cost of communication throughput [6]. This makes sense in a context where most applications collect small pieces of data, such as temperature, humidity or lighting measurements. However, modern applications are now being deployed to gather acoustic and visual data, which have a high demand for communication throughput. Energy consumption remains a key consideration in the design of WSNs, but throughput may be equally important in these new kinds of applications.

This change of paradigm has motivated researchers to develop new platforms for WSNs aimed to deliver higher throughput while still being energy efficient. One technique that allows this goal to be achieved is to provide radio diversity, which means having more than one radio in each WSN mote. If these radios operate in different frequency bands, they will not interfere with each other while communicating, reducing packet loss due to interference and allowing simultaneous transmissions between sensor nodes. Radio diversity can significantly improve end-to-end delivery rates, network stability and transmission costs while incurring a small increase in energy cost over a single radio [7].

Many multi-radio platforms have been proposed in the literature for WSNs [8–11] and for Internet of Things (IoT), including: the IoT DevKit-LoRaWAN [12], Multi-Transceiver consisting of LoRa and ESP8266-Wifi Communication Module [13], Wi-Fi and LoRa radios [14]; Pycom's FiPy [15], which is a device with multiples radios for LoRa, Sigfox, WiFi, and Bluetooth; Dual-radio motes, such as Waspmote [16], OpenMote B [17], and Firefly [18], which has 2.4 GHz short-range and 920 MHz long-range radios; Narrowband Internet of Things (NB-IoT) (an emerging cellular technology) and 802.15.4 for sink nodes [19]. For this work, we adopted the Opal mote [6], which includes two radios, one operating in the 900 MHz band and the other in the 2.4 GHz band with the ability to transmit or receive packets through both radios at the same time.

Such capability was exploited in the design of FastForward [20], a high-throughput protocol for dual-radio WSNs where each intermediate

* Corresponding author.
E-mail addresses: nildo@dcc.ufmg.br (N.d.S. Ribeiro Junior), mmvieira@dcc.ufmg.br (M.A.M. Vieira), lfvieira@dcc.ufmg.br (L.F.M. Vieira), gnawali@cs.uh.edu (O. Gnawali).

https://doi.org/10.1016/j.comnet.2022.108832
Received 25 March 2020; Received in revised form 17 November 2021; Accepted 30 January 2022
Available online 18 February 2022

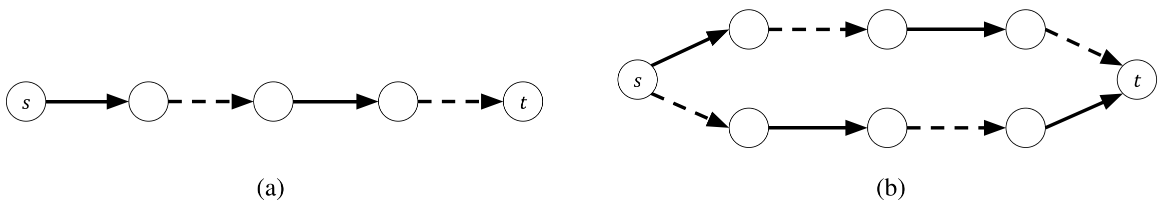


Fig. 1. Forwarding scheme where each intermediate node receives packets through one radio and sends packets through another.

node in the path receives packets through one of the radios and transmits through the other, as it is shown in Fig. 1(a). With this scheme, it is possible to double the maximum throughput one would be able to achieve using single radio platforms. However, notice that although the intermediate nodes use both radios, the source and the destination nodes use only one of them, thus allowing the source or destination to participate in transmission or reception of just one data stream at a time.

The main idea of this work is to fully utilize all the available radios, even those in the source and destination nodes, to achieve the maximum throughput we can using dual-radio platforms. In order to accomplish this goal, we can calculate and use two disjoint paths, where each intermediate node will receive packets through one radio and retransmit them through the other one, just as the previous scheme, but the source node will be using both radios for transmissions and the destination node will be using both for receiving packets, as it is show in Fig. 1(b).

Using two disjoint paths to transmit packets potentially doubles the throughput compared with the single path scheme. However, it adds a new constraint to the routing problem. The two paths must have lengths of the same parity. If this is not the case, there will be a bottleneck in the network, where the destination node would be receiving packets coming from two different paths through the same radio, and the result would be the same as having a single path.

In this paper, we present the design, implementation and evaluation of SplitPath, a distributed routing protocol that computes two paths with the same parity to achieve high end-to-end throughput in a dual-radio wireless sensor network. SplitPath is the first protocol to use multiple paths in dual-radio WSNs to achieve maximum throughput, and the first routing algorithm aiming to find multiple paths with the constraint of having the same parity.

Our main contributions are: (i) the definition of a new relevant problem in digraphs with practical application in routing for wireless sensor networks; (ii) proof that the problem is NP-complete; (iii) a model for an optimal centralized solution using integer linear programming; (iv) a polynomial distributed solution; (v) experiments in the real world, achieving 96% of the theoretical limit, which is twice of throughput achieved by the state-of-the-art protocols.

The organization of this paper is as follows. In Section 2 we discuss the related work. In Section 3 we formalize the problem and detail the theoretical basis. In Section 4 we present a centralized solution for the problem. In Section 5 we present the SplitPath decentralized algorithm to solve the problem. In Section 6 we describe the implementation of our protocol. In Section 7 we summarize the experiments we did to evaluate the performance of our protocol and compare it with the state-of-the-art protocol. Finally, in Section 8 we present the conclusion of this work.

2. Related work

2.1. High throughput in WSNs

Protocols developed for traditional applications in WSNs do not achieve high throughput. Their design prioritized minimizing the total energy consumption in each node of the network, to guarantee a longer lifetime of these nodes. As WSN applications that required high throughput were proposed, researchers proposed many protocols specific for bulk data transfer, such as Flush [21], FlushMF [22] and PIP [23]. These protocols use a single path route between a source node and a destination, disable duty-cycling and reserve the radio channels until the transfer is complete. They try to achieve high throughput by enforcing an end-to-end schedule of packet transmissions across each hop. These protocols consider the problem of achieving high throughput to be orthogonal to the low-power operation. A technique called Burst Forward [24] allows the intermediate nodes to keep their duty-cycle and achieve the same throughput as the previous protocols. It achieves its goal by using a combination of high-throughput bursts and a store-and-forward method.

However all of these protocols are limited to a maximum theoretical end-to-end throughput of 50% of the channel capacity, which is a fundamental problem of single radio platforms. Furthermore, to eliminate self-interference these protocols use channel hopping, and the overhead of changing channels decreases the throughput such that the best performing protocols only achieve one quarter of the channel capacity [23].

Other works suggested enhancing the network by adopting dual radio platforms. H. Li et al. [19] shows that the performance of 802.15.4-Based WSN can be enhanced with additional emerging cellular Narrowband Internet of Things (NB-IoT) radio at the sink nodes. SEDA-Net [25] proposes to use dual radio platforms, where there is a long-range and a short-range radio. One radio is used as the wake-up radio that acts as a sentinel and the other radio is used for data transmission. The configuration of which long or short-range radio should be used as sentinel is defined dynamically. But, these works do not focus on throughput maximization.

With the development of dual-radio platforms for WSNs, such as Opal [6], the total throughput could be improved and theoretically reach 100% of the channel capacity. FastForward [20] is the main protocol in the literature to explore the advantages of using dual-radio platforms in order to maximize throughput in a WSN. FastForward transmits packets from a source node to a destination through a single path. The radio used to transmit the packets along the path are alternated in each hop. It assumes that one radio does not interfere with the other, which already helps to solve the problem of intra-path interference. Furthermore, it uses multiple fixed channels for each radio in each hop, thereby further reducing the effect of interference and eliminating the overhead of switching channels along the way.

2.2. Multipath routing algorithms

Various multipath routing algorithms for WSNs in the literature aim to solve different problems. Some routing algorithms consider the problem of finding exactly two disjoint paths in a network. The algorithms presented by Ishida et al. [26] and Ogier and Shacham [27] are designed to find a pair of disjoint paths from each node to a single destination. Lee and Reddy [28] present an algorithm to find two paths that are as disjoint as possible and the second path is as short as possible. Griffin and Korkmaz [29] present an algorithm to check if the network contains at least two disjoint paths between all node pairs.

Another problem in many multipath routing algorithms is to find k disjoint paths between a source and a destination node. Beginning with the centralized algorithms to approach this problem, there are those that will try to find any set of k disjoint paths and guarantee correctness [30–32][33]. Other centralized algorithms find the k disjoint paths with minimum total length and guarantee correctness and optimality [34–37][38]. Chuzhoy et al. [39] showed that node disjoint path problem is $2^{\Omega(\sqrt{\log n})}$-hard to approximate. The main problem of using such centralized algorithms in WSNs is the necessity of collecting every node's neighbor list to be aware of the whole topology, making the communication cost very high for large networks.

Distributed algorithms for a general k were also proposed in the literature [40–47]. These algorithms are efficient, but do not try to minimize the length of the routing paths. They find the paths one by one, and once a path is found, it is fixed and its internal nodes are removed from the network, so they cannot be selected to form another path. On the other hand, Zhang et al. [48] present a distributed algorithm for finding k disjoint paths with minimum length in WSNs called OFDP. Unlike the other decentralized algorithms, OFDP does not fix the paths it finds. Instead, it marks the nodes as occupied and considers the edges to be only backwards and with negative weights. Then, in each iteration it searches for augmenting paths in the network. OFDP finds k disjoint paths between a source and a destination with minimum total length in the kth round, if they exist.

The multiple path communication has been studied extensively in Wireless Mesh Networks [49–52] but they assume radios are homogeneous and do not require the path parity constraint.

Thus, none of the algorithms cited in this section consider the problem of finding disjoint paths with the same parity.

3. Disjoint parity paths problem

Each node has two distinct radios that can operate simultaneously. Utilizing both radios in each node, it is possible to maximize data transmission from source to destination if two paths are found. Those paths should obey two conditions.

First, they should be disjoint, except the common source and destination nodes. All intermediate nodes should belong to, at most, one path. If a node is chosen for two paths, it will not be able to receive and transmit the packets from the two flows on two paths simultaneously, creating a bottleneck in the network and losing the benefits of using two radios.

The second condition is that both paths should have the same number of hops parity. We need length of both the paths to be either odd or even; mixing paths with odd and even path lengths is not allowed. This is because the source needs to transmit packets by different radios and the destination needs to receive the packets also by different radios, so that both can operate simultaneously. Fig. 2 shows the reasoning for this condition. The figure shows a network with several nodes, and possibly two disjoint paths between the source node s and the final destination t. The paths chosen in (a) have different parity, one has an odd number of hops and the other has even. Since the source node needs to transmit by different radios and the intermediate nodes need to alternate their radios to send, the destination ends up receiving packets from both packets in the same radio, which is impossible to occur simultaneously. Differently, the path chosen in (b) has the same parity. For this reason, the destination node ends up receiving the packets by two distinct radios, which can happen at the same time.

Given a digraph $D = (V, A)$, the disjoint parity paths problem is to find two simple paths P_1 and P_2, starting at the same source node s and ending in the same destination node t, with all intermediate nodes disjoint, with both paths having the same number of hops parity, i.e., $|P_1| \mod 2 = |P_2| \mod 2$.

We model the problem as a digraph because for generic dual-radio platforms the radios are usually not the same, since the objective is to provide radio diversity. For many different radios, range or channel capacity varies a lot. Some platforms might even use directional antennas. Furthermore, the problem formulation as a digraph also accounts for asymmetric links that happen in the real world and are also observed on testbeds.

To deal with the interference, we allocated a different channel for each link in its interference region. There are 10 available channels at 900 Mhz and 16 more channels at 2.4 GHz. We intend to investigate when the number of channels is insufficient in the near future.

3.1. NP-completeness

The even path in digraphs problems is stated as follows: Given a digraph $D = (V, A)$ and $s, t \in V$, is there an even-length simple path from s to t? This problem is shown to be NP-complete by LaPaugh and Papadimitriou [53]. Using this result, we can show the following:

Theorem 1. *Given a digraph $D = (V, A)$ and $s, t \in V$, it is NP-complete to decide whether there are two simple paths from s to t, with all intermediate vertices disjoint, and lengths of the same parity.*

Proof. Fist of all, the disjoint parity paths problem is in NP as we can check a solution for the problem by counting the number of arcs in each path, comparing their parity and verifying that each intermediate vertex appears only once, and this can be done in polynomial time.

Next, we will reduce the even path in the digraphs problem to the disjoint parity path problem. Given a digraph $D = (V, A)$ and $s, t \in V$, we construct a new digraph $D' = (V', A')$ as follows:

$$V' = V \cup \{s', t', n\}$$

$$A' = A \cup \{(s', s), (t, t'), (s', n), (n, t')\}$$

Basically, we add three new vertices and four new arcs, as shown in Fig. 3, and this transformation is $O(1)$. We can see that D' is constructed in a way that it already has one simple path of even length from s' to t' going through n. The other path must include the arcs (s', s) and (t, t'), and it will be of even length if, and only if, there is a path of even length from s to t. Therefore, D' will have two disjoint simple paths of the same parity from s' to t' if, and only if, D has an even-length simple path from s to t. $\square$

If the problem of finding two even paths is NP-complete, then the general problem of finding paths with the same parity is already NP-complete, whether finding two paths of odd lengths is polynomial or exponential. However, a similar reduction can be done in the case of two odd paths. We just need to add two new vertices, one in each path and outside the original graph. This way, each path will be of odd length if, and only if, D has an even-length simple path from s to t. Then the problem of finding two paths with the same parity in digraphs is NP-complete whether the desired parity is even or odd.

4. Centralized solution

In previous section, we showed that the existence problem is NP-Complete. In this section, we solve a different version of the same problem. We consider the minimization of sum lengths and we model the problem using integer linear programming. The complete model is shown in Equation 1.

We represent the network as a directed graph $D = (V, A)$. For each arc (i, j) of the graph, we associate two different weights, c^1_{ij} and c^2_{ij}, each represents the cost to send one message from vertex i to vertex j. c^1_{ij} is the cost to send a message via radio 1 and c^2_{ij} is the cost to send via radio 2. For each arc, we also define two binary variables, x^1_{ij} and x^2_{ij}, that have value 0 if the arc (i, j) is not chosen for a path, or value 1, in case that arc is chosen. x^1_{ij} represents that radio 1 is chosen and x^2_{ij} represents that radio 2 is chosen. The goal to minimize the total cost, that is mapped by the sum of the cost of each radio multiplied by the correspondent x variable.

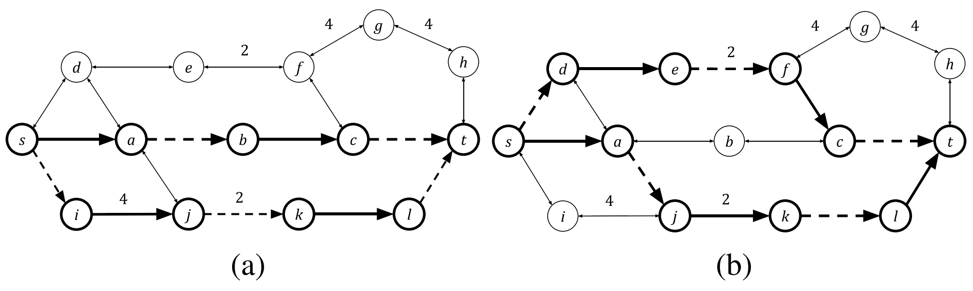


Fig. 2. Example of two disjoint paths in a network. In (a) with different parity and in (b) with the same parity.

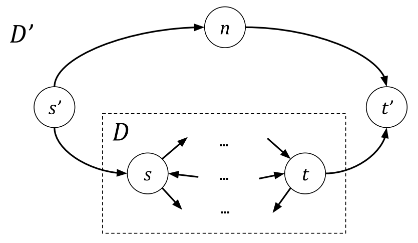


Fig. 3. Reducing the even path in digraphs problem to the disjoint parity paths problem.

To model the constraints we need to formalize some definitions: we denote s the source node and t the destination node. For each vertex $i \in V$, we define two sets of arcs: $S(i)$ is the set of arcs leaving node i, and $E(i)$ is the set of arcs entering node i.

The constraints (i) and (ii) guarantee that the radio alternates between the chosen paths. In constraint (i), for each vertex i, the sum of the arcs leaving radio 1 minus the sum of the arcs entering by radio 2 must be 1 if the node is the source, −1 if it is the destination, or 0 for all the other cases. This guarantees that the source node will not receive by radio 2 and for sure will send through radio 1. We also guarantee that the destination will receive by radio 2 and will not send by radio 1. For all the other nodes, if it receives by radio 2, it will definitely send by radio 1.

$$\text{minimize} \sum_{(i,j)} c^1_{ij} x^1_{ij} + c^2_{ij} x^2_{ij}$$

subject to

$$\text{(i)} \quad \sum_{(i,j)\in S(i)} x^1_{ij} - \sum_{(j,i)\in E(i)} x^2_{ji} = \begin{cases} 1, \text{ if } i = s \\ -1, \text{ if } i = t \\ 0, \text{ otherwise} \end{cases}$$

$$\text{(ii)} \quad \sum_{(i,j)\in S(i)} x^2_{ij} - \sum_{(j,i)\in E(i)} x^1_{ji} = \begin{cases} 1, \text{ if } i = s \\ -1, \text{ if } i = t \\ 0, \text{ otherwise} \end{cases}$$

$$\text{(iii)} \quad \sum_{j\in E(i)} x^1_{ji} + \sum_{j\in E(i)} x^2_{ji} \leq 1, \quad \text{if } i \neq t$$

$$\text{(iv)} \quad \sum_{(i,j)\in A} x^1_{ij} - \sum_{(i,j)\in A} x^2_{ij} = 0$$

$$\text{(v)} \quad x^1_{ij}, x^2_{ij} \in \{0,1\}$$

Equation 1: Integer linear programming model to solve the shortest dual path parity routing problem.

Similarly to previous constraint, in constraint (ii) for each vertex i, the sum of the arcs that leave radio 2 minus the sum of the arcs that enter by radio 1 must be 1 for the source, −1 for the destination and 0 for the other cases. This guarantee that the source node will not receive by radio 1 neither and for sure will send by radio 2. The destination node will for sure receive by radio 1 and will not send by radio 2. For all the other nodes, if it receives by radio 1, it will send via radio 2.

Constraint (iii) guarantees that no intermediate node will be chosen for two paths, making the sum of arcs entering in all node i, except the destination node, be less or equal to 1. As the variables are binary, either radio 1 is chosen and radio 2 is not, or radio 2 is chosen and radio 1 is not, or none of them is chosen. For the destination node, two entering arcs are chosen, one for each radio, as guarantee constraints (i) and (ii). Constraints (i) and (ii) also guarantee that we do not need to create a constraint for the number of leaving arcs for each node, as it must be equal to the number of entering arcs in each intermediate node.

Finally, constraint (iv) is the one that guarantees that both paths will have the same parity. It says that the number of arcs where radio 1 is chosen is equal to the number of arcs where radio 2 is chosen. Since the nodes always alternate the radios in the paths, it enforces them to have the same parity. Constraint (v) determines that variables x^1_{ij} and x^2_{ij} are binary.

5. Distributed algorithm

SplitPath works by finding a shortest path between s and t of a certain parity, and then trying to find an augmenting path of the same parity of the one that was found before. A pseudo code of the main procedure is shown in 1. It has two main procedures, called FindPath and TracePath and two phases.

In the first phase, the algorithm tries to find a path and an augmented path of even length. It does that by enforcing the source node to transmit its first FIND message through a specific radio, and the destination node to accept paths that result in a FIND message received through the other radio. Intermediate nodes always alternate the radios they receive and send messages. We use the notation $\bar{r}$ to indicate the opposite radio, so, if a node receives through r, it sends through $\bar{r}$.

Similarly, in the second phase, the algorithm tries to find a path and an augmented path of odd length, enforcing the destination node to only accept FIND messages from the same radio r that the source node used to transmit its first FIND message. Trying to find a path in the network that has a certain parity induce an odd cycle in some path. To avoid cycles, the FIND packets include the path until node v, and if v is in the path, it just ignores the message.

A node may be marked as free or occupied. Being marked as a free node means that it is not a part of any current paths. At the beginning, all nodes start marked as free nodes, and becomes occupied if it is chosen to be a part of a path in a TRACE step. If a node is occupied, it knows the nodes that are its predecessor and the successor in the path, in variables we called $prev(v)$ and $next(v)$.

Let v be any node in D. We assume that v knows all its neighbors and the weights of the incident arcs. The header of each message contains its sender's ID, so the receiver is able to know where an incoming message is coming from. In the next subsections we describe the algorithms for each procedure of SplitPath.

Algorithm 1 SplitPath(D, s, t)

```
    Input: direct graph D, source node s, destination node t
    Output: two disjoint paths s-t with same parity
 1:                                   ▷ First phase (Even length)
 2: FindPath($s, t, 0$)                ▷ Path starting through radio 0
 3: TracePath($s, t, 1$)               ▷ and ending through radio 1
 4: if a path was found then
 5:     FindPath($s, t, 1$)
 6:     TracePath($s, t, 0$)
 7:     if two paths were found then
 8:         paths0 ← getPaths($D$)
 9:     end if
10: end if
11:                                   ▷ Second phase (Odd length)
12: FindPath($s, t, 0$)                ▷ Path starting through radio 0
13: TracePath($s, t, 0$)               ▷ and ending through radio 0
14: if a path was found then
15:     FindPath($s, t, 1$)
16:     TracePath($s, t, 1$)
17:     if two paths were found then
18:         paths1 ← getPaths($D$)
19:     end if
20: end if
21:
22: return min(paths0, paths1)
```

5.1. Finding an augmenting path

The pseudo code for the FindPath procedure is presented in Fig. 4. Each node v maintains variables $d_r(v)$ and $p_r(v)$. $d_r(v)$ records the length of the shortest path from s to v such that the FIND message was received by v through radio r, and $p_r(v)$ records the associated path. If it is an occupied node, it also stores variables $Bd_r(v)$ and $Bp_r(v)$ that represent the distance and paths coming from back arcs in a selected path.

The nodes try to find an augmenting path by exchanging FIND messages. Each FIND message sent by u via radio r has a field $d(u)$, which is the length of the shortest path from s to u that arrived in u via radio $\bar{r}$ found until now. The other field in the FIND message, $p(u)$, is the node list that represents the shortest path associated with $d(u)$.

At the beginning of a FindPath procedure, s broadcasts a FIND message with $d(s) = 0$ and $p(s) = \{\}$ via radio r, which is defined in the main procedure of SplitPath, to start the process. When a node $v \neq s$ receives a FIND message $\{d(u), p(u)\}$ from u via radio r, there are three possible cases we discuss next:

Case 1: v is a free node. We use a variable $dist(v)$ to record the length from s to v via radio r. We let $dist(v)$ be $d(u) + l_r(u, v)$, where $l_r(u, v)$ is the cost of sending a message from u to v via radio r. If $dist(v) < d_r(v)$, it means that the path that just arrived via radio r is shorter than the shortest path found until now, then we let this be the new shortest path from s to v via radio r. Also, if the path is updated, v broadcasts a FIND message $\{d_r(v), p_r(v)\}$ via radio $\bar{r}$.

Case 2: v is an occupied node and u is not $prev(v)$ or $next(v)$. This means v received a message from a free node, thus the arc (u, v) is not part of an established path. We let $dist(u)$ be $d(u)+l_r(u, v)$, and compare with the current shortest path. If the shortest path is updated and $v \neq t$, v sends a FIND message $\{d_r(v), p_r(v)\}$ just to $prev(v)$, otherwise it would allow an augmenting path to cross an occupied node.

Case 3: v is an occupied node and u is $next(v)$. The later condition means that the arc (v, u) is part of a selected path using radio $\bar{r}$. Then, we let $dist(v)$ be $d(u) - l_{\bar{r}}(u, v)$, to allow our path to go through this back arc, and eliminate it if this augmenting path is chosen. For this case, we need to store the distance and the path in the variables $Bd_r(v)$ and $Bp_r(v)$, respectively. Then, v broadcasts a FIND message $\{d_r(v), p_r(v)\}$ via radio $\bar{r}$.

```
 1: procedure FindPath($s, t, r$)
 2:     Input: source node $s$, destination node $t$, radio $r$
 3:     Compute: disjoint path $s - t$ starting from radio $r$
 4:     $s$ broadcasts $\{0, \{\}\}$ via radio $r$
 5:     while $v$ receives $\{d(u), p(u)\}$ via $r$ do
 6:         case $v = s$ or $v$ is in $p(u)$:
 7:             Ignore message
 8:         case $v$ is a free node:
 9:             $dist(v) \leftarrow d(u) + l_r(u, v)$
10:             if $dist(v) < d_r(v)$ then
11:                 $d_r(v) \leftarrow dist(v)(v)$
12:                 $p_r(v) \leftarrow p(u) \cup \{u\}$
13:                 Broadcast $\{d_r(v), p_r(v)\}$ via $\bar{r}$
14:             end if
15:         case $v$ is an occupied node and $u$ is not $prev(v)$
                    and $u$ is not $next(v)$:
16:             $dist(v) \leftarrow d(u) + l_r(u, v)$
17:             if $dist(v) < d_r(v)$ then
18:                 $d_r(v) \leftarrow dist(v)$
19:                 $p_r(v) \leftarrow p(u) \cup \{u\}$
20:                 if $v \neq t$ then
21:                     Send $\{d_r(v), p_r(v)\}$ to $prev(v)$ via $\bar{r}$
22:                 end if
23:             end if
24:         case $v$ is an occupied node and $u$ is $next(v)$:
25:             $dist(v) \leftarrow d(u) - l_{\bar{r}}(u, v)$
26:             if $dist(v) < Bd_r(v)$ then
27:                 $Bd_r(v) \leftarrow dist(v)$
28:                 $Bp_r(v) \leftarrow p(u) \cup \{u\}$
29:                 Broadcast $\{Bd_r(v), Bp_r(v)\}$ via $\bar{r}$
30:             end if
31:     end while
32: end procedure
```

Fig. 4. FindPath procedure.

At each time a node v broadcasts a message with a path including n hops, its neighbors will have a path of $n + 1$ hops. The maximum number of hops of an augmenting path in the FindPath procedure will be three times the diameter of the network in number of hops, which would be a case where the path goes back almost all the back arcs. Therefore, the time complexity of this procedure is $O(diam)$, where $diam$ is the diameter of the network. About the number of messages, each node sends a message when it is updated, and it can be updated if any of its predecessors is updated. Therefore, in the worst case, the communication complexity is in the order of O($diam^2$) messages.

5.2. Tracing an augmenting path

The TracePath pseudocode is shown in Fig. 5. This procedure is started by t. When t receives its first FIND message in the previous FindPath procedure, it waits for a long enough time to start the execution of TracePath. Each TRACE message has one field $p_r(t)$, which is the list of nodes in the shortest augmenting path that arrived in t via radio r. The radio r is a parameter defined by the main procedure of SplitPath.

Node t starts by sending a TRACE message to its predecessor in the chosen path. When a node v receives a TRACE message from u, we define two new variables $succ(v)$ and $pred(v)$, that receives the successor and the predecessor of v in the received path $p(t)$.

```
1: procedure TracePath($s, t, r$)
2:     Input: source node $s$, destination node $t$, radio $r$
3:     Compute: Trace disjoint path $s - t$ starting from radio $r$
4:     $pred(t) \leftarrow$ predecessor of $t$ in $p_r(t)$
5:     $t$ sends $\{p_r(t)\}$ to $pred(t)$ via $r$
6:     while $v \neq s$ receives a new $\{p(t)\}$ via $r$ do
7:         $succ(v) \leftarrow$ successor of $v$ in $p(t)$
8:         $pred(v) \leftarrow$ predecessor of $v$ in $p(t)$
9:         case $v$ is a free node:
10:            Mark itself as an occupied node
11:            $prev(v) \leftarrow pred(v)$
12:            $next(v) \leftarrow succ(v)$
13:        case $v$ is an occupied node
                and $succ(v)$ is not $prev(v)$:
14:            Still mark itself as an occupied node
15:            $next(v) \leftarrow succ(v)$
16:        case $v$ is an occupied node and $succ(v)$ is $prev(v)$
                and $pred(v)$ is $next(v)$:
17:            Mark itself as a free node
18:        case $v$ is an occupied node and $succ(v)$ is $prev(v)$
                and $pred(v)$ is not $next(v)$:
19:            Still mark itself as an occupied node
20:            $prev(v) \leftarrow pred(v)$
21:        Send $\{p(t)\}$ to $pred(v)$ via $\bar{r}$
22:    end while
23: end procedure
```

Fig. 5. TracePath procedure.

Case 1: v is a free node. This is a trivial case. It just marks itself as an occupied node and sets its $prev(v)$ and $next(v)$ to $pred(v)$ and $succ(v)$, respectively.

Case 2: v is an occupied node and $succ(v)$ is not $prev(v)$. This means that node v is the last hop of a backward segment. It is the case of node a in the execution example of Fig. 6(b). In this case, node v changes its $next(v)$ variable to $succ(v)$, and keep its old $prev(v)$.

Case 3: v is an occupied node and $succ(v)$ is $prev(v)$ and $pred(v)$ is $next(v)$. It means that node v is in the middle of a backward segment, illustrated by node b in Fig. 6(b). v marks itself as a free node, since it will not be a part of any path anymore.

Case 4: v is an occupied node and $succ(v)$ is $prev(v)$ and $pred(v)$ is not $next(v)$. In this case, v is the first node of a backward segment, as node c in the example of Fig. 6(b). In this case, node v needs to change its $prev(v)$ variable to $pred(v)$.

After evaluating and executing the proper case, every node in the path that received a TRACE message must retransmit the message to its predecessor. In the standard case, the communication complexity of the TracePath procedure will be $O(diam)$ messages, where $diam$ is the diameter of the network. But the TRACE message has to be delivered to every node in the selected path. In a directed graph, any arc in the selected path might not have a back arc. When an arc in a selected path (u, v) does not have a back arc (v, u), the simplest way to work around it is v starts broadcasting the TRACE message until it gets to u. If this is necessary, the communication complexity of TracePath will be $O(V)$ in number of messages in the worst case. The time complexity will always be $O(diam)$.

5.3. An example of SplitPath's execution

Fig. 6 shows an example execution of SplitPath. Consider that each arc without a value assigned has weight 1. Also, all the arcs have the same weight in both ways.

Fig. 6(a) shows the result of the first round of FindPath and TracePath procedures. It finds a path starting through radio 0 (continuous arrow) and ending through radio 1 (dashed arrow). Now each of these arcs (u, v) will be removed, and the back arc (v,u) through the opposite radio will be assigned with a negative weight equal of the removed arc.

In Fig. 6(b) is the result of the second round of FindPath procedure. As in the previous round s started sending through radio 0 and t received through radio 1, in this round, s starts sending through radio 1 and t accepts paths received through radio 0. The first path t will receive through radio 0 will be $\{s, d, e, f, g, h, t\}$, with total weight of 13. But when c receives a FIND message from f, it will follow case 2 and send a message to b. Then b will be in case 3, store the path coming from a back arc and broadcast a path with length $5-1=4$. Node a will receive the message through another back arc, also fall in case 3, and broadcast a path with length $4-1=3$. It will be received by j, and from it will continue until it gets to t through radio 0 with length 8. Then t will update its path to the path $\{s, d, e, f c, b, a, j, k, l, t\}$, with contains two back arcs.

Finally, in Fig. 6(c) is the result of the second round of TracePath procedure. The TRACE message will make the free nodes l, k, j, f, e and d fall in case 1, which is straightforward. Node a will execute case 2 and change $next(a)$ to j in the new path. Then node b will execute case 3 and mark itself as a free node and not be part of any path anymore. And node c will be on case 4, and change $prev(c)$ to f in the augmenting path. We will finish the algorithm with two same parity paths with total length of 12.

6. Implementation

SplitPath was implemented using TinyOS 2.1.2 for the Opal platform [6]. This platform has two 802.15.4 radio transceivers that operate in different bands: 900 MHz and 2.4 GHz. The two radios share the same SPI bus, which creates a bottleneck for data transfer between the radios and the micro-controller. However, data transfer on the bus is much faster than data transmission over the radio. On the Opal platform, sending an SPI packet to the transmission buffer takes less than 10% of the time it takes to transmit the packet over the radio [20]. As the protocol seeks to keep the radios always busy, the two radios will be operating at the same time most of the time.

Fig. 7 shows an overview of the architecture implemented for SplitPath. It was implemented over the communication protocol stack for TinyOS: the radio driver, and the implementation of the MAC and link layers. It offers options for configuring the modulation used by radios, transmit power, Clear Channel Assessment (CCA) on/off, and random back-off parameters used. There is still the possibility of enabling or disabling the use of acknowledgment packets provided by the link layer.

7. Evaluation

7.1. Quality of the solution

To the best of our knowledge, there is no other work in the literature solving the minimum parity disjoint paths problem, so there is no proper baseline to compare our heuristic with. We decided to compare our heuristic to another heuristic consisting of using an existing distributed algorithm to solve the minimum k disjoint paths problem, OFDP [48]. To find two paths with the same parity with OFDP, we run it with setting $k = 3$. The algorithm will find three disjoint paths in the network with the minimum total weight, therefore, it is guaranteed that at least two of them will have the same parity. Then we choose the pair of paths with same parity and smallest total weight. We chose this algorithm to compare our results because it is how we can use an existing distributed algorithm and guarantee that we are going to find paths with the same parity with it. Also, as we have shown, if this algorithm finds an answer, it is guaranteed that ours will also find one.

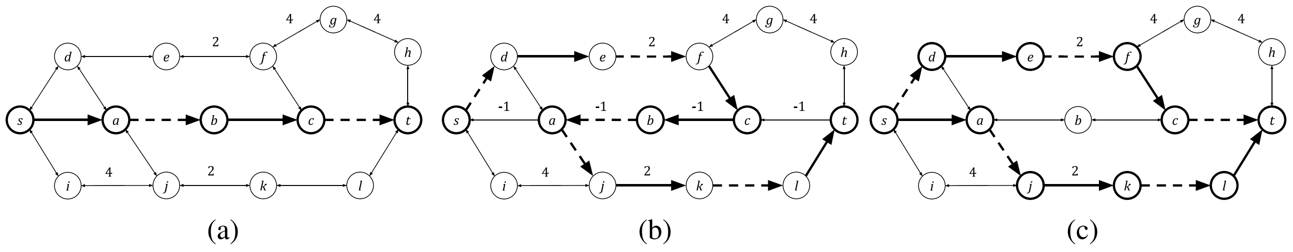


Fig. 6. Example of SplitPath's execution.

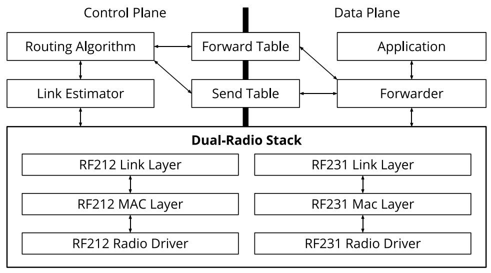


Fig. 7. Implemented architecture.

We show that our results are much better than using OFDP for this purpose.

We generated five random instances of a network topology, each one with different densities, increasing the number of edges in the graph from 10% to 90% of the total number of possible edges, at equal steps of 20%. In each instance we ran the ILP solution to find the optimal, and run both algorithms for every combination of two nodes as source and destination, such that source and destination are not adjacent.

In Fig. 8(a) we can see the comparison of our heuristic with OFDP($k = 3$) in number of optimal solutions found. Our solution found over 80% of optimal solutions for every tested network density, slightly increasing as the density increases. With OFDP($k = 3$), the optimal solutions goes from 32% to 49%. In Fig. 8(b) we show the average additional cost from the non-optimal solutions. Our heuristic's greatest additional cost was about 7% of the optimal in less dense networks. The smallest average additional cost with OFDP($k = 3$) was 11% in more dense networks, but reaching up to 22% at 10% density.

7.2. Throughput evaluation

We conducted experiments on a large-scale wireless sensor network testbed that contains 100 sensor nodes with two radios from the Opal platform. The experiments evaluated throughput and data yield using SplitPath. Several rounds of experiments were performed, and each round consisted of the following steps: (i) determination of source and destination nodes, (ii) execution of SplitPath routing algorithm to establish the two routes and (iii) transmission of data from the source node to the destination.

We configured the two radios to transmit using the O-QPSK modulation at 250 kbps and with transmission power of 3 dBm. In the first experiments, we left enabled, in the MAC layer, the Clear Channel Assessment (CCA) and the random back-offs. Later, these functions were disabled to compare with FastForward results. We also performed the experiments with and without acknowledgment packets, to analyze the compromise between the throughput and the data yield in both cases.

In each experiment, the source node sends 1000 packets. Each packet has 127 bytes, where the payload has 100 data bytes. We utilized up to 100 sensor nodes in these experiments. We define the throughput as the total number of bytes received by the destination node per second, including those not related to the payload in the packet. We define the data yield as the number of unique packets received by the destination node divided by the total number of packets sent by the source node. The experiments were repeated 10 times for each instance, and the values presented are the mean and standard deviation of the obtained results.

The maximum transmission rate, considering an ideal environment, using only one radio is 250 kbps, or 31.25 kBps. We represent this with a green dashed line. The theoretical limit for two radios is twice or 62.5 kBps. This is represented with a continuous green line.

To deal with problems caused by interference, we assign different channels for each hop in the paths, and in our experiments in the real world we have results that are accounting the interference happening in a realistic scenario.

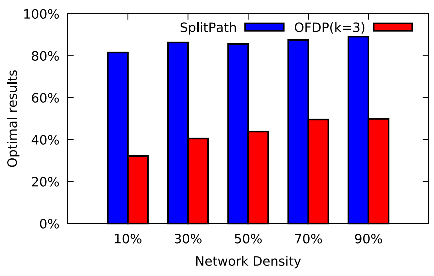
We present results that compare the performance of the SplitPath protocol against our FastForward implementation. We do not compare to CTP [54], XCTP [55] or RPL because they are for single-radio. For this case, they are worse than FastForward, which is the state-of-the-art for dual-radio. Fig. 9 shows the result with CCA enabled. Top figure shows the throughput and bottom shows the data yield. We can observe that SplitPath achieved higher throughput. On average, SplitPath achieved a 60% throughput improvement in this scenario. The data yield of both protocols were similar, achieving 100% data yield on most cases.

Fig. 10 shows the performance of the protocols when we disable the CCA in the MAC layer of the radios. In practice, it is not recommended to disable this function because the media can be used by several different networks and one can congestion the other. The test was performed to evaluate the maximum transmission potential of the protocols. Acknowledgment packets are also used in this scenario. We can observe that in the first cases, SplitPath achieves 50 kBps throughput, while FastForward reaches a maximum of 25 kBps, an improvement of 100%.

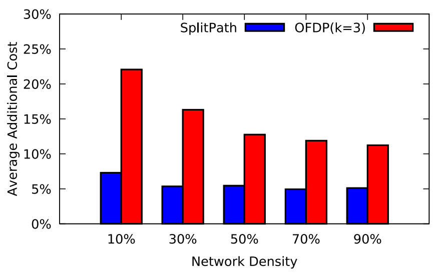
Finally, Fig. 11 represents a scenario where both the CCA and acknowledgment have been disabled. In this scenario, we obtained the maximum throughput of 60 kBps. This value is close to the maximum theoretical throughput limit when using two radios. Therefore, the value obtained is 96% of ideal theoretical maximum flow and it was obtained in a real environment. In addition, this value is twice the value achieved with FastForward, indicating the maximum use of the two radios in intermediate nodes. The data rate did not reach 100% because of interference and packet error since CCA and ACK were disabled. But this scenario is important to show that SpliPath could reach 96% of ideal theoretical maximum flow in practice.

7.3. Path quality evaluation

We also evaluated the path quality of the solutions. The experiments were done in 5700 instances of the testbed. The metric used for the link quality is the Expected Transmission Count (ETX) [56], which indicates the number of expected transmissions of a packet necessary for it to be received without error at its destination. For a one hop link, it can be measured as the ratio of number of packets sent to the number of

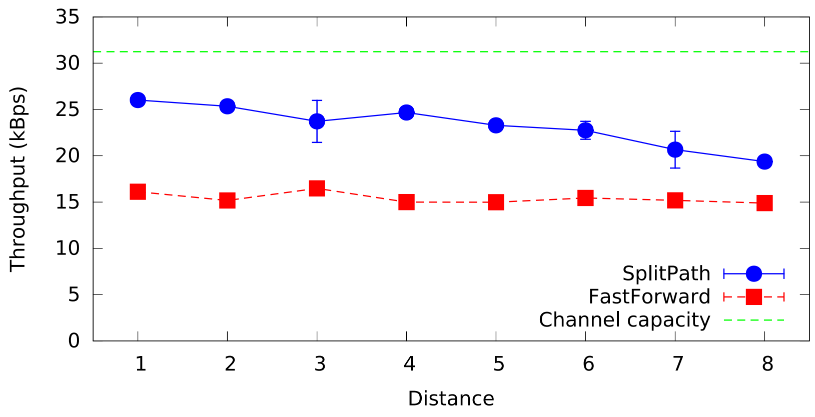


(a) Percentage of instances with optimal results

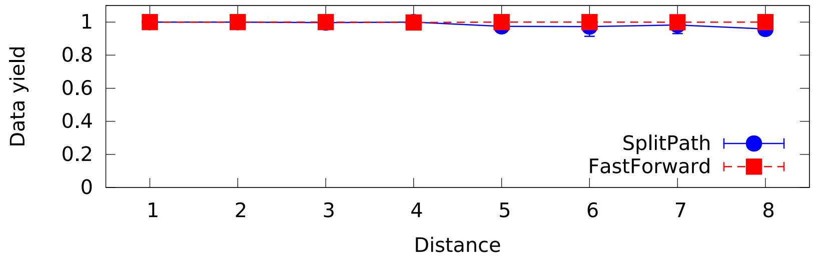


(b) Average additional cost in non-optimal results

Fig. 8. Comparing the results for our heuristic and OFDP($k = 3$) in random network topologies with different densities.



(a) Throughput



(b) Data yield

Fig. 9. Performance when CCA is enabled and ACK is required.

packets received, which is the inverse of the packets delivery rate. The larger the number of packets received, the smaller is the cost, given that the number of packets sent stays the same. The cost of a path is defined as the sum of the costs of each link used in the path [56]. If all the paths are 100% reliable (which means ETX = 1) then longer paths will have higher ETX cost.

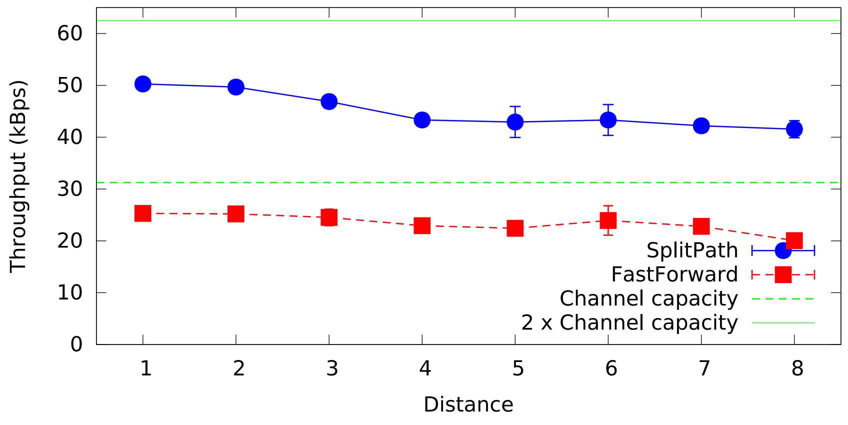
Fig. 12 shows the Cumulative Distribution Function (CDF) of the cost of the paths for all 5700 solutions. Path number 2 (darker color) is the path with higher cost. The figure shows that the maximum cost found by SplitPath was 135. It also illustrates the difference between the first and second path. Remember that SplitPath computes paths with the same parity, but not necessarily with the same size. Fig. 12 enables us to visualize this difference in path cost.

7.4. Direct or undirect edge?

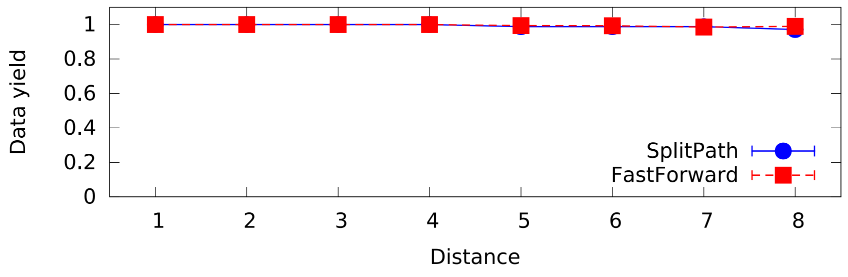
SplitPath is designed to work with a direct graph D. In case someone wonders if it is necessary to handle direct links and not just undirect links, we also executed experiments in the testbed to verify how often the links are asymmetrics.

We collected the complete topology of the testbed and represented it in a digraph, where $w(u,v)_r$ represents the cost of transmitting a packet from u to v through radio r. We uploaded a program where each node broadcast a total of 100 packets through each radio, with a random interval between 1 and 2 s to each transmission. When a node v receives a packet from node u through radio r, it adds u to a list of neighbors where each entry has two counters, one for each radio, and increments the counter for radio r. At the end of the experiment, each node v has a counter $c(u,v)_r$ for the number of messages received from each neighbor u through radio r. The metric we use to assign the weights of each link in the testbed topology is the Packet Error Rate (PER), which is a rate on the number of packets lost in each link. u will only be in the neighbors' table of v if at least one packet is received. If no message was received through a certain link, the digraph model of the network does not contain the related edge.

Now we analyze the symmetry in the testbed digraph model. In Fig. 13 we can see a histogram of the weight differences between each pair of edges $(u,v)_r$ and $(v,u)_r$. We also show in a red line the Cumulative Distribution Function. For each of these pairs of edges, we

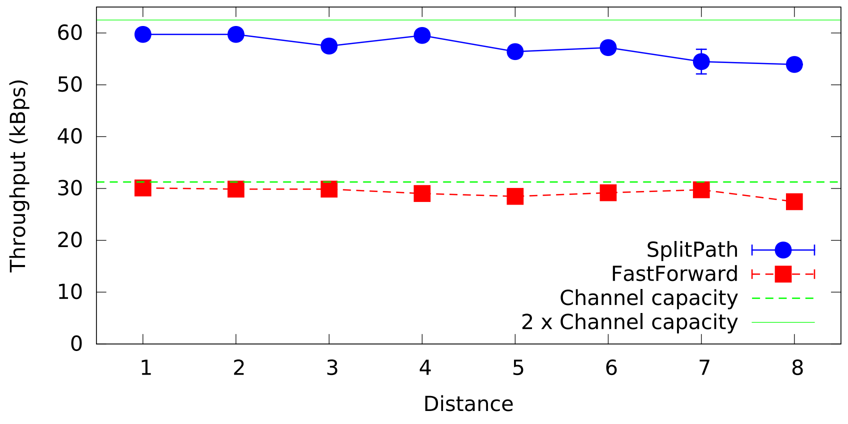


(a) Throughput

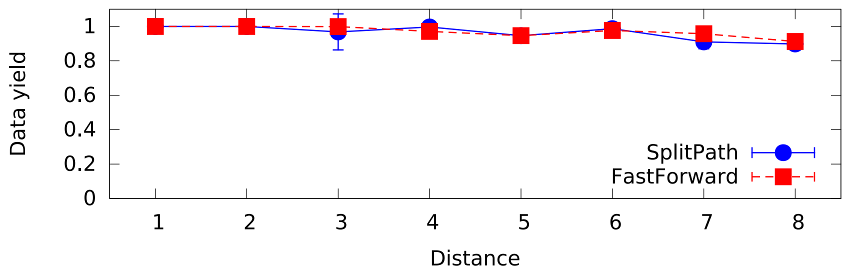


(b) Data yield

Fig. 10. Performance when CCA is disabled and ACK is required.



(a) Throughput



(b) Data yield

Fig. 11. Performance when CCA is disabled and ACK is not required.

calculate the absolute value of $w(u,v)_r - w(v,u)_r$ and count the number of edges with weight differences in intervals of 10. We can see that about 60% of the pair of edges have a weight difference smaller than 10. Only 4% of the links are completely symmetric, most of the partially symmetric links have a small weight difference. Furthermore, 7.4% of the edges have a weight difference greater than 30.

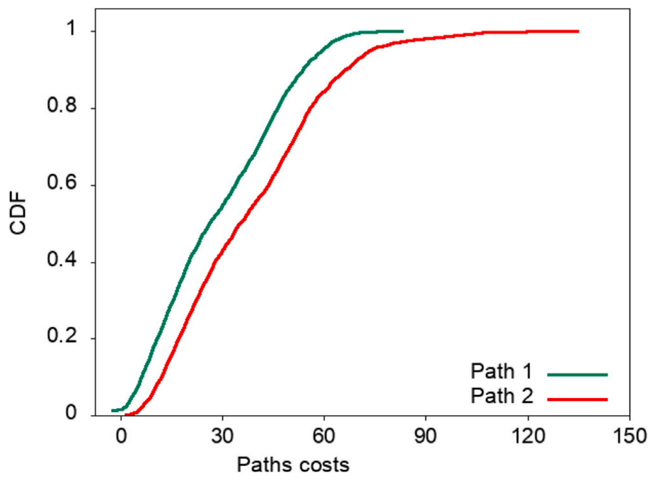


Fig. 12. Cumulative Distribution Function of the cost of each path found by SplitPath.

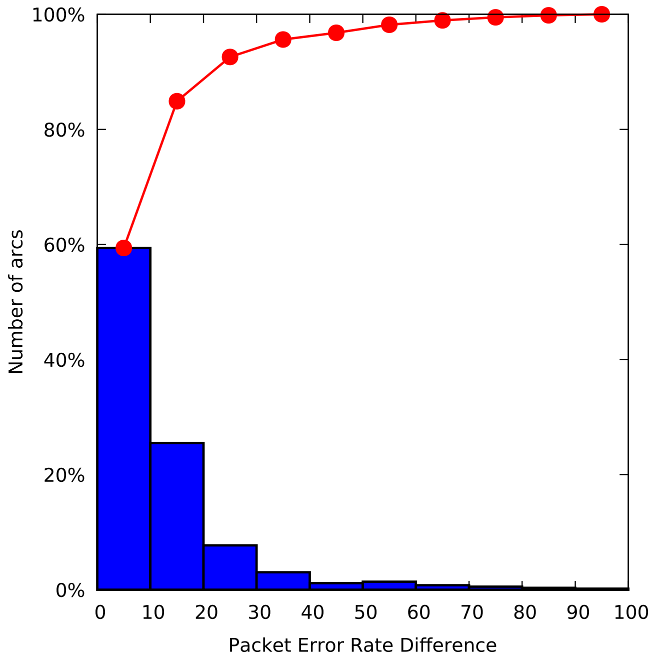


Fig. 13. Symmetry distribution in the testbed. Only 60% of the links have a PER difference smaller than or equal to 10%. Thus, direct links are a reality.

Thus, we verified experimentally that the testbed presents many direct edges. Fortunately, SplitPath is designed to handle direct links.

7.5. *Energy consumption*

About energy consumption, with a larger throughput, it is expected the total energy expenditure to increase. But the energy consumption per transmitted byte becomes smaller. According to Jurdak [6], the Opal consumes an average of 49 mA of current if both radios are operating simultaneously. As our protocol reached a flow rate of 60 kBps, we have an energy expenditure of 0.82 mA/kB. While with the maximum flow reached by FastForward of 30 kBps, we would have an energy expenditure of 1.64 mA/kB. Thus, our protocol consumes half of the energy per byte transmitted on each node. Because the number of nodes using two paths does not double, since the source and destination nodes are always the same, our protocol will have a greater overall energy efficiency than FastForward.

8. Conclusion

In this work, we presented SplitPath, a new protocol for bulk-data transfer optimized for dual-radio platforms. The main advantage of this new approach is to utilize the two transceivers available on each node in parallel on the source, intermediate, and destination nodes. The protocol aims to meet the demand for high throughput in new applications of wireless sensor networks that need to transmit multimedia data in real time.

Experiments carried out in the physical world show that the proposed approach achieves throughput of up to 60 kBps, which represents 96% of the theoretical maximum limit of 62.5 kBps when using two 802.15.4 radios with O-QPSK modulation at 250 kbps, without checking channel occupancy. For an application that needs to share the communication medium and checks if the channel is busy, our protocol achieves throughput of up to 26 kBps vs. 16 kBps of the current state-of-the-art FastForward protocol, a 60% performance improvement.

For future work, we consider finding an approximation algorithm for the disjoint parity paths problem. Our current approach gives excellent practical results, but it would be good to know provable bounds. We also want to study the problem of extending the protocol for multiple radios and simultaneous multiple sources and destinations.

Finally, we would like to combine SplitPath with a routing protocol, such as CGR [57], that considers the energy consumption of the sensor node to reroute some paths and also maximize the lifetime of the network.

CRediT authorship contribution statement

Nildo dos Santos Ribeiro Junior: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Marcos A.M. Vieira:** Conceptualization, Methodology, Resources, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration, Funding acquisition. **Luiz F.M. Vieira:** Conceptualization, Methodology, Resources, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration, Funding acquisition. **Omprakash Gnawali:** Conceptualization, Methodology, Resources, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would also like to thank the research agencies Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, CNPq, Brazil, FAPESP, Brazil 2018/23085-5 and FAPEMIG, Brazil for their financial support.

References

[1] M.A.M. Vieira, C.N. Coelho, D.C. da Silva, J.M. da Mata, Survey on wireless sensor network devices, in: EFTA 2003. 2003 IEEE Conference on Emerging Technologies and Factory Automation. Proceedings, Vol. 1, Cat. No.03TH8696, 2003, pp. 537–544, http://dx.doi.org/10.1109/ETFA.2003.1247753.
[2] I.F. Akyildiz, M.C. Vuran, Wireless Sensor Networks, Vol. 4, John Wiley & Sons, 2010.
[3] L.B. Ruiz, L.H.A. Correia, L.F.M. Vieira, D.F. Macedo, E.F. Nakamura, C.M. Figueiredo, M.A.M. Vieira, E.H.B. Maia, D. Câmara, A.A. Loureiro, et al., Architectures for wireless sensor networks, in: Proceedings of the 22nd Brazilian Symposium on Computer Networks, SBRC'04, 2004, pp. 167–218.

[4] B.P. Santos, O. Goussevskaia, L.F. Vieira, M.A. Vieira, A.A. Loureiro, Mobile matrix: Routing under mobility in IoT, IoMT, and social IoT, Ad Hoc Netw. 78 (2018) 84–98, http://dx.doi.org/10.1016/j.adhoc.2018.05.012, URL https://www.sciencedirect.com/science/article/pii/S1570870518302415.

[5] J. Yick, B. Mukherjee, D. Ghosal, Wireless sensor network survey, Comput. Netw. 52 (12) (2008) 2292–2330.

[6] R. Jurdak, K. Klues, B. Kusy, C. Richter, K. Langendoen, M. Brünig, Opal: A multiradio platform for high throughput wireless sensor networks, IEEE Embedded Syst. Lett. 3 (4) (2011) 121–124.

[7] B. Kusy, C. Richter, W. Hu, M. Afanasyev, R. Jurdak, M. Brünig, D. Abbott, C. Huynh, D. Ostry, Radio diversity for reliable communication in wsns, in: Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on, IEEE, 2011, pp. 270–281.

[8] J. Beutel, Fast-prototyping using the BTnode platform, in: Design, Automation and Test in Europe, 2006. DATE'06. Proceedings, Vol. 1, IEEE, 2006, pp. 1–6.

[9] M. Kohvakka, T. Arpinen, M. Hännikäinen, T.D. Hämäläinen, High-performance multi-radio WSN platform, in: Proceedings of the 2nd International Workshop on Multi-Hop Ad Hoc Networks: From Theory to Reality, ACM, 2006, pp. 95–97.

[10] A. Burns, B.R. Greene, M.J. McGrath, T.J. O'Shea, B. Kuris, S.M. Ayer, F. Stroiescu, V. Cionca, SHIMMER^{TM}–a wireless sensor platform for noninvasive biomedical research, IEEE Sensors J. 10 (9) (2010) 1527–1534.

[11] E. Popovici, D. Boyle, S. O'Connell, S. Faul, P. Angove, J. Buckley, B. O'Flynn, J. Barton, C. O'Mathúna, The s-Mote: A versatile heterogeneous multi-radio platform for wireless sensor networks applications, in: Circuit Theory and Design (ECCTD), 2011 20th European Conference on, IEEE, 2011, pp. 421–424.

[12] L. Alliance, IoT DevKit - LoRaWAN, 2021, https://lora-alliance.org/.

[13] N. Azmi, S. Sudin, L. Munirah Kamarudin, A. Zakaria, R. Visvanathan, G. Chew Cheik, S.M. Mamduh Syed Zakaria, K. Abdullah Alfarhan, R. Badlishah Ahmad, Design and development of multi-transceiver lorafi board consisting LoRa and ESP8266-Wifi communication module, in: Materials Science and Engineering Conference Series, in: Materials Science and Engineering Conference Series, vol. 318, 2018, 012051, http://dx.doi.org/10.1088/1757-899X/318/1/012051.

[14] D.H. Kim, J.Y. Lim, J.D. Kim, Low-power, long-range, high-data transmission using Wi-Fi and LoRa, in: 2016 6th International Conference on IT Convergence and Security, ICITCS, 2016, pp. 1–3, http://dx.doi.org/10.1109/ICITCS.2016.7740351.

[15] P. Ferreira, R.N. Miranda, P.M. Cruz, H.S. Mendonça, Multi-protocol LoRaWAN/Wi-Fi sensor node performance assessment for industry 4.0 energy monitoring, in: 2019 IEEE-APS Topical Conference on Antennas and Propagation in Wireless Communications, APWC, 2019, pp. 403–407, http://dx.doi.org/10.1109/APWC.2019.8870468.

[16] libelium, Waspmote, 2021, http://www.libelium.com/products/waspmote.

[17] P. Tuset, X. Vilajosana, T. Watteyne, OpenMote B, 2021, https://openmote.com/product/openmote-b/.

[18] Zolertia, Firefly mote, 2021, http://zolertia.io/product/hardware/firefly.

[19] H. Li, W. Dong, Y. Wang, Y. Gao, C. Chen, Enhancing the performance of 802.15.4-based wireless sensor networks with NB-IoT, IEEE Internet Things J. 7 (4) (2020) 3523–3534, http://dx.doi.org/10.1109/JIOT.2020.2972512.

[20] G. Ekbatanifard, P. Sommer, B. Kusy, V. Iyer, K. Langendoen, Fastforward: High-throughput dual-radio streaming, in: Mobile Ad-Hoc and Sensor Systems, 10th International Conf. on, IEEE, 2013, pp. 209–213.

[21] S. Kim, R. Fonseca, P. Dutta, A. Tavakoli, D. Culler, P. Levis, S. Shenker, I. Stoica, Flush: a reliable bulk transport protocol for multihop wireless networks, in: Proceedings of the 5th International Conference on Embedded Networked Sensor Systems, ACM, 2007, pp. 351–365.

[22] R. Tavares, M.A.M. Vieira, L.F.M. Vieira, FlushMF: A transport protocol using multiple frequencies for wireless sensor network, in: Proceedings of the 13th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, MASS, IEEE, 2016, pp. 192–200.

[23] B. Raman, K. Chebrolu, S. Bijwe, V. Gabale, PIP: A connection-oriented, multi-hop, multi-channel TDMA-based MAC for high throughput bulk transfer, in: Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems, ACM, 2010, pp. 15–28.

[24] S. Duquennoy, F. Österlind, A. Dunkels, Lossy links, low power, high throughput, in: Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems, ACM, 2011, pp. 12–25.

[25] J. Jung, J. Hong, Y. Yi, On self-configuring IoT with dual radios: A cross-layer approach, IEEE Trans. Mob. Comput. (2021) 1, http://dx.doi.org/10.1109/TMC.2021.3066441.

[26] K. Ishida, Y. Kakuda, T. Kikuno, A routing protocol for finding two node-disjoint paths in computer networks, in: Network Protocols, 1995. Proceedings., 1995 International Conference on, IEEE, 1995, pp. 340–347.

[27] R. Ogier, N. Shacham, A distributed algorithm for finding shortest pairs of disjoint paths, in: INFOCOM'89. Proceedings of the Eighth Annual Joint Conference of the IEEE Computer and Communications Societies. Technology: Emerging Or Converging, IEEE, IEEE, 1989, pp. 173–182.

[28] Y.O. Lee, A.N. Reddy, Disjoint multi-path routing and failure recovery, in: Communications, IEEE International Conference on, IEEE, 2010, pp. 1–6.

[29] M. Griffin, T. Korkmaz, Distributed verification of global multiple disjoint paths in mobile wireless networks, in: Wireless Communications and Mobile Computing Conference (IWCMC), 7th International, IEEE, 2011, pp. 666–670.

[30] S. Khuller, B. Schieber, Efficient parallel algorithms for testing k and finding disjoint s-t paths in graphs, SIAM J. Comput. 20 (2) (1991) 352–375.

[31] K. Iwama, C. Iwamoto, T. Ohsawaa, A faster parallel algorithm for k-connectivity, Inform. Process. Lett. 61 (5) (1997) 265–269.

[32] Y.-q. Chen, X.-f. Guo, Q.-k. Zeng, G.-h. CHEN, Amr: a multipath routing algorithm based on maximum flow in Ad-Hoc networks, Acta Electron. Sin. 32 (8) (2004) 1297–1301.

[33] R. Grossi, A. Marino, L. Versari, Efficient algorithms for listing k disjoint st-paths in graphs, in: Latin American Symposium on Theoretical Informatics, Springer, 2018, pp. 544–557.

[34] A. Srinivas, E. Modiano, Minimum energy disjoint path routing in wireless ad-hoc networks, in: Proceedings of the 9th Annual International Conference on Mobile Computing and Networking, ACM, 2003, pp. 122–133.

[35] R. Bhandari, Optimal physical diversity algorithms and survivable networks, in: Computers and Communications, 1997. Proceedings., Second IEEE Symposium on, IEEE, 1997, pp. 433–441.

[36] J. Suurballe, Disjoint paths in a network, Networks 4 (2) (1974) 125–145.

[37] T. Hashiguchi, K. Tajima, Y. Takita, T. Naito, Node-disjoint paths search in WDM networks with asymmetric nodes, in: 15th International Conference on Optical Network Design and Modeling, IEEE, 2011, pp. 1–6.

[38] A.A. de Melo, C.M.H. de Figueiredo, U.d.S. Souza, On undirected two-commodity integral flow, disjoint paths and strict terminal connection problems, Networks 77 (4) (2021) 559–571, http://dx.doi.org/10.1002/net.21976, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.21976.

[39] J. Chuzhoy, D.H.K. Kim, R. Nimavat, New hardness results for routing on disjoint paths, SIAM J. Comput. http://dx.doi.org/10.1137/17M1146580, STOC17–189–STOC17–237.

[40] D. Ganesan, R. Govindan, S. Shenker, D. Estrin, Highly-resilient, energy-efficient multipath routing in wireless sensor networks, ACM SIGMOBILE Mob. Comput. Commun. Rev. 5 (4) (2001) 11–25.

[41] S. Li, Z. Wu, Node-disjoint parallel multi-path routing in wireless sensor networks, in: Embedded Software and Systems, 2005. Second International Conference on, IEEE, 2005, p. 6.

[42] J.W. Baek, Y.J. Nam, D.-W. Seo, An energy-efficient k-disjoint-path routing algorithm for reliable wireless sensor networks, in: IFIP International Workshop on Software Technolgies for Embedded and Ubiquitous Systems, Springer, 2007, pp. 399–408.

[43] D. Sidhu, R. Nair, S. Abdallah, Finding disjoint paths in networks, SIGCOMM Comput. Commun. Rev. 21 (4) (1991) 43–51.

[44] X. Fang, S. Shi, J. Li, A disjoint multi-path routing algorithm in wireless sensor network, J. Comput. Res. Dev. 46 (12) (2009) 2053–2061.

[45] S. Omar, M. Zoulikha, B. Cousin, Energy efficiency in ad hoc wireless networks with node-disjoint path routing, in: Systems, Signal Processing and their Applications (WOSSPA), 2011 7th International Workshop on, IEEE, 2011, pp. 127–130.

[46] A. Kumar, S. Varma, Geographic node-disjoint path routing for wireless sensor networks, IEEE Sens. J. 10 (6) (2010) 1138–1139.

[47] R. Nicolescu, H. Wu, BFS solution for disjoint paths in p systems, in: C.S. Calude, J. Kari, I. Petre, G. Rozenberg (Eds.), Unconventional Computation, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 164–176.

[48] K. Zhang, Q. Han, G. Yin, H. Pan, OFDP: a distributed algorithm for finding disjoint paths with minimum total length in wireless sensor networks, J. Comb. Optim. 31 (4) (2016) 1623–1641.

[49] A. Raniwala, T. cker Chiueh, Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network, in: Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies, Vol. 3, 2005, pp. 2223–2234, http://dx.doi.org/10.1109/INFCOM.2005.1498497.

[50] M.K. Marina, S.R. Das, A.P. Subramanian, A topology control approach for utilizing multiple channels in multi-radio wireless mesh networks, Comput. Netw. 54 (2) (2010) 241–256.

[51] R. Draves, J. Padhye, B. Zill, Routing in multi-radio, multi-hop wireless mesh networks, in: Proceedings of the 10th Annual International Conference on Mobile Computing and Networking, ACM, New York, NY, USA, 2004, pp. 114–128.

[52] M. Alicherry, R. Bhatia, L.E. Li, Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks, in: Proceedings of the 11th Annual International Conference on Mobile Computing and Networking, ACM, New York, NY, USA, 2005, pp. 58–72.

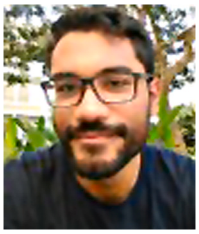
[53] A.S. LaPaugh, C.H. Papadimitriou, The even-path problem for graphs and digraphs, Networks 14 (4) (1984) 507–513.

[54] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, P. Levis, Collection tree protocol, in: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems, ACM, New York, NY, USA, 2009, pp. 1–14.

[55] B.P. Santos, M.A.M. Vieira, L.F.M. Vieira, eXtend collection tree protocol, in: 2015 IEEE Wireless Communications and Networking Conference, WCNC, 2015, pp. 1512–1517, http://dx.doi.org/10.1109/WCNC.2015.7127692.

[56] D.S.J. De Couto, High-Throughput Routing for Multi-Hop Wireless Networks (Ph.D. thesis), Massachusetts Institute of Technology, 2004.

[57] B.P. Santos, L.F. Vieira, M.A. Vieira, CGR: Centrality-based green routing for low-power and lossy networks, Comput. Netw. 129 (2017) 117–128, http://dx.doi.org/10.1016/j.comnet.2017.09.009, URL https://www.sciencedirect.com/science/article/pii/S1389128617303602.



Nildo dos Santos Ribeiro Junior received his M.Sc. in Computer Science at the Universidade Federal de Minas Gerais (UFMG). His research interests are in Wireless Sensor Networks and Computer Networking.

Marcos A.M. Vieira is an Associate Professor of the Computer Science Department at the Universidade Federal de Minas Gerais (UFMG). He received his undergraduate and M.S. at the Universidade Federal de Minas Gerais in Belo Horizonte, and M.S. and Ph.D. degrees in Computer Science from the University of Southern California (USC). His research interests are in Computer Networking and Wireless Networks.

Luiz F.M. Vieira is an Associate Professor of the Computer Science Department at the Universidade Federal de Minas Gerais (UFMG). He received his undergraduate and M.S. at the Universidade Federal de Minas Gerais in Belo Horizonte, and the Ph.D. degree in Computer Science from the University of California Los Angeles (UCLA). His research interests are in Computer Networking and Wireless Networks.

Omprakash Gnawali is an Assistant Professor at the University of Houston, USA. He was a Postdoctoral Scholar at Stanford University, got his Ph.D. from the University of Southern California, and received his Masters and Bachelors degrees from the Massachusetts Institute of Technology. His research lies at the intersection of low power wireless networks and embedded sensing systems.