Distrib. Comput. (2001) 14: 217–229

DISTRIBUTED COMPUTING

Interval routing schemes allow broadcasting with linear message-complexity

Pierre Fraigniaud[1,*], **Cyril Gavoille**[2,**], **Bernard Mans**[3,***]

[1] Laboratoire de Recherche en Informatique, Bât. 490, Université Paris-Sud, 91405 Orsay cedex, France
[2] Laboratoire Bordelais de Recherche en Informatique, Université Bordeaux I, 33405 Talence cedex, France
[3] Department of Computing, Division of ICS, Macquarie University, Sydney, NSW 2109, Australia

Received: December 2000 / Accepted: July 2001

Summary. The purpose of *compact routing* is to provide a labeling of the nodes of a network and a way to encode the routing tables, so that routing can be performed efficiently (e.g., on shortest paths) whilst keeping the memory-space required to store the routing tables as small as possible. In this paper, we answer a long-standing conjecture by showing that compact routing may also assist in the performance of distributed computations. In particular, we show that a network supporting a shortest path *interval routing scheme* allows broadcasting with a message-complexity of $O(n)$, where n is the number of nodes of the network. As a consequence, we prove that $O(n)$ messages suffice to solve leader-election for any graph labeled by a shortest path interval routing scheme, improving the previous known bound of $O(m+n)$. A general consequence of our result is that a shortest path interval routing scheme contains ample *structural information* to avoid developing ad-hoc or network-specific solutions for basic problems that distributed systems must handle repeatedly.

Key words: Compact routing – Interval routing – Broadcasting – Distributed computing

1 Introduction

This paper addresses a problem originally formulated by D. Peleg that can be informally summarized as follows: "Do networks supporting shortest path compact routing schemes present specific ability in term of distributed computation? E.g., broadcasting, leader election, etc." This paper answers this question in the affirmative, by showing that n-node networks supporting interval routing schemes [27,28] (IRS for short), allow broadcasting with $O(n)$ message-complexity.

Formally, a network $G=(V,E)$ (in this paper, by *network*, we will always mean a connected undirected graph without self loops and multi-edges) supports an IRS if the nodes of that network can be labeled from 1 to $n=|V|$ in such a way that the following is satisfied: given any node $x \in V$ of degree d and label ℓ, there is a set of d intervals $I_1,\ldots,I_d$ of $\{1,\ldots,n\}$, one for each edge $e_1,\ldots,e_d$ incident to x, such that (1) $\{1,\ldots,n\} \setminus \{\ell\} \subseteq \bigcup_{i=1}^{d} I_i$, (2) $I_i \cap I_j = \emptyset$ for every $i \neq j$ and (3) $\ell' \in I_i$ implies that there is a shortest path from x to the node labeled ℓ' passing through the edge e_i. (IRS encode shortest path routing tables with the property that the set of destination-addresses using a given link is a set of consecutive integers.) IRS are well-known in the framework of compact routing since a network of maximum degree Δ, and supporting an IRS, has routing tables of size $O(\Delta \log n)$ bits, in comparison to the $\Theta(n \log \Delta)$ bits of a table returning, for every destination label $i \in \{1,\ldots,n\}$, the output port corresponding to that label. For more about IRS, we refer to [7,11, 18,20,23,24], and to the survey [17]. For more about compact routing in general, we refer to [13–15,19,21,26]. In the remainder of this paper, given a network supporting an IRS, we will make no distinction between the nodes and their labels. In other words, we will assume $V=\{1,\ldots,n\}$ where the label of node x in the IRS is precisely $x \in \{1,\ldots,n\}$.

Broadcasting for an arbitrary node of a network is the information dissemination problem which consists of sending a given message to all the other nodes. The message-complexity of broadcasting is between $\Omega(n)$ and $O(m)$, $m=|E|$, since the reception of the message by every node but the source requires at least $n-1$ messages, and yet, broadcasting can always be performed by flooding the network (meaning, upon reception of the message, every node forwards that message through all its incident edges apart from the one through which it has received the message). Improved upper and lower bounds may be derived as a function of the knowledge of the nodes of the network and of the maximal size of the message-headers (e.g., see [1,2,5]). In this paper, the only knowledge of every node is its label in some IRS and the intervals attached to its incident edges in the same IRS. The size of each message-header

Part of this work was completed while the third author was visiting the Computer Science Department of University Paris-Sud at LRI, supported by the Australian-French ARC-IREX/CNRS cooperation #99N92/0523. A preliminary version of this paper was presented at the 19th ACM Symposium on Principles of Distributed Computing (PODC 2000).

* Additional support by the CNRS
** Additional support by the Aquitaine Region project #98024002
*** Additional support by the ARC

transmitted by our broadcast protocol is $\lceil \log_2 n \rceil + 1$ bits. (In respect to the time-complexity of broadcasting, we refer the reader to [3,12,22].)

The relationship between IRS and broadcasting has been previously investigated. Van Leeuwen and Tan [28] proved that minimum spanning tree construction (and therefore broadcasting) and other related distributed problems such as leader-election, may be solved by exchanging $O(n)$ messages in a ring whose nodes are labeled according to an IRS and $O(m+n)$ messages for arbitrary graphs whose nodes are labeled according to an IRS. Let us recall that leader-election without any network knowledge requires $\Theta(n \log n)$ messages for a ring and $\Theta(m+n \log n)$ messages for an arbitrary graph [16]. More generally, the question of how much a labeling can help in the solution of distributed problems was studied in [9,10] in the framework of Sense of Direction [8]. It was shown in [6] that the message-complexity of the broadcast problem is $n-1$ for the restricted class of networks supporting *all-shortest-path* IRS (an IRS where all the shortest paths are represented) with additional restrictions on the intervals (strictness and linearity). Finally, de la Torre, Narayanan and Peleg [4] showed that the same result holds in IRS networks satisfying the so-called ssr*-tree* property. (Informally, this property states that, for any node x, the set of paths induced by the IRS originated at x and ending at all the other nodes, is a tree.) In this paper, we improve these results by showing that a network supporting standard shortest path IRS supports a broadcast protocol of message-complexity $\Theta(n)$.

Our result has many consequences on other problems such as *leader-election* or *distributed spanning tree*. For instance, Korach et al. [25] have shown that the leader-election problem may be solved using $(b(n) + n)(\log_2 n + 1)$ messages where $b(n)$ is the message-complexity of broadcasting in an n-node network. This result, combined with our own, shows that n-node networks supporting shortest path IRS allow the leader-election problem to be solved with $O(n \log n)$ message-complexity. In fact, we will prove, by giving a specific algorithm based on our $O(n)$-message broadcast protocol, that $O(n)$ messages suffice to solve leader-election for any graphs whose nodes are labeled according to a shortest path IRS. This improves the $O(m+n)$ previous bound of van Leeuwen and Tan [28]. Note that the set of networks supporting IRS forms a wide class of graphs including, e.g., unitary interval and circular-arc graphs, hypercubes, multi-dimensional tori (see [17]).

This paper is organized according to several hypotheses on the IRS. These hypotheses will be relaxed as the paper proceeds. We distinguish two types of intervals: an interval $[a,b] = \{a,\ldots,b\}$ with $b \geq a$ is said to be *linear*, whereas an interval $[a,b]$ with $b < a$ refers to the set $\{a,\ldots,n,1,\ldots,b\}$ and is said to be *cyclic*. The class of networks supporting linear IRS (LIRS for short), those in which all intervals of the IRS are linear, is strictly included in the class of networks supporting IRS. We also distinguish the case $V \setminus \{x\} = \bigcup_{i=1}^{d} I_i$ for every node x[1], from the case in which x appears in the interval of one of its incident edges, for at least one node x. In the former case, the IRS is said to be *strict*. Hence we get four types of interval routing schemes: IRS, LIRS, strict IRS, and strict LIRS. The following section presents some preliminary results. Section 3 is dedicated to networks supporting strict LIRS. Section 4 relaxes the strictness requirement. Section 5 relaxes the linearity requirement and presents our main result about IRS. Section 6 contains some remarks about related problems, specifically leader election and distributed spanning tree computation. Finally, Sect. 7 contains some concluding remarks about possible extensions based on the results of this paper.

[1] Recall that we assumed $V = \{1,\ldots,n\}$ and we make no distinction between the nodes and their labels.

2 Preliminary results

In this section, we will present a distributed broadcast protocol for a network supporting an IRS. This protocol is simple though efficient, since its message-complexity will be shown to be $O(n)$. We refer to this protocol as the up/down protocol. The second part of the section will present tools for the analysis of this protocol.

2.1 The up/down broadcast protocol

Let ν be the source of the broadcast, identified by its label in the IRS. The source initiates the broadcast by sending two copies of the message, one meant for the node $\nu - 1$ and the other meant for the node $\nu + 1$. The latter copy is called the "up" copy and the former the "down" copy. (If $\nu = 1$ or $\nu = n$, then the source sends only one copy.) There will be at most two copies of the message circulating in the network. First we concentrate on the up copy meant for $\nu + 1$. The message will eventually reach $\nu + 1$ by the shortest path set by the IRS. It may possibly cross intermediate nodes but these nodes will simply forward the message to its destination, disregarding of its content. Once $\nu + 1$ receives the message, it reads its content, modifies the header by replacing $\nu + 1$ by $\nu + 2$ and forwards it towards the node labeled $\nu + 2$. More generally, once a node labeled x, $\nu < x < n$, receives the message meant for x, it reads its content, modifies the header by replacing x by $x + 1$ and forwards it toward node labeled $x + 1$. When the node labeled n receives the message, it reads its content and removes it from the network. The same strategy is applied for the down copy, by replacing $x + 1$ by $x - 1$, $1 < x < \nu$, until it reaches the node labeled 1, which reads its content and removes it from the network. Again, at every given time, a copy of the message is meant for only one specific node, called the *target*, and when a node different from that target receives the message, it does not take the opportunity to read its content, but just forwards the message to the target along the shortest path set by the IRS leading to that target.

The up/down protocol uses headers of size $\lceil \log_2 n \rceil + 1$ bits: the label of the destination has $\lceil \log_2 n \rceil$ bits and there is one bit indicating whether it is a down copy or an up copy. The message-complexity of the up/down protocol is $\sum_{i=1}^{n-1} d(i, i+1)$ where $d(x,y)$ is the distance between the node labeled x and the node labeled y in G. To the knowledge of the authors, no improved bounds on $d(i, i+1)$ for networks supporting IRS are known. Note that it is known [6] that $d(i, i+1) = 1$ for networks supporting an *all-shortest-paths* strict LIRS (that is for networks such that a strict LIRS encodes *all* shortest paths). The class of networks supporting all-shortest-paths strict LIRS

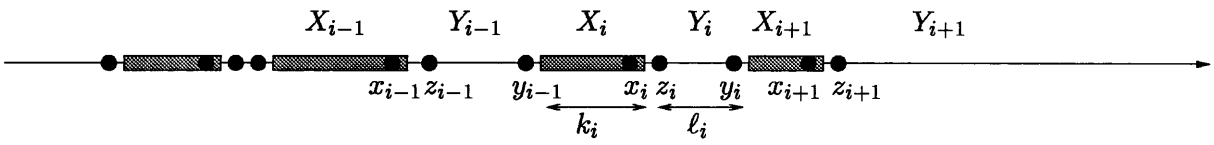

Fig. 1. Notations

is very restricted and, as we will see, only weaker results may be proved for (single shortest path) LIRS or IRS.

Assume, for the sake of clarity, that the source is the node labeled 1 (the general case will be considered in later sections). Let $W = w_1, w_2, \ldots, w_\tau$ be the sequence of nodes visited by the message from its source w_1 (labeled 1) to its final destination w_τ (labeled n). The same node may appear several times in W, however, a node x can only appear once as a target node. The possible other occurrences of node x correspond to steps of the protocol in which x is traversed by a message meant for a target node $y \neq x$. Let us complement the sequence W by two virtual nodes, $w_0 = n + 1$ and $w_{\tau+1} = 0$. These two nodes are not target nodes. More precisely, W can be written as:

$$W = Y_0, X_1, Y_1, X_2, Y_2, \ldots, X_{\varrho-1}, Y_{\varrho-1}, X_\varrho, Y_\varrho \qquad (1)$$

where X_i, $i = 1, \ldots, \varrho$, is a maximal sequence of consecutive target nodes and Y_i, $i = 1, \ldots, \varrho - 1$, is the sequence of non-target nodes between the last node of X_i and the first node of X_{i+1}. We set $Y_0 = w_0$, and $Y_\varrho = w_{\tau+1}$. The nodes of the Y_i's are called *intermediate* nodes.

Notation. For every i, throughout the paper, we will always make use of the following notation (see Fig. 1).

- $k_i = |X_i|$ and $\ell_i = |Y_i|$;
- $x_i =$ the last node of X_i;
- $y_i =$ the last node of Y_i; and
- $z_i =$ the first node of Y_i.

2.2 Intermediate node sequences

The next four lemmas give some properties satisfied by the intermediate nodes in the Y_i's. These properties will be shown to be helpful in computing the message complexity of the up/down protocol.

Lemma 1 *Let G be a network supporting a strict LIRS. Let $u_0 = x, u_1, \ldots, u_k = x + 1$ (resp. $v_0 = x, v_1, \ldots, v_m = x - 1$) be the shortest path from node x to node $x + 1$ (resp. to node $x - 1$) set by the strict LIRS. Then*

$$u_1 > u_2 > \ldots > u_{k-1} > x + 1.$$

(Respectively, $v_1 < v_2 < \ldots < v_{m-1} < x - 1$.)

Proof. Let I_i be the interval of edge (u_i, u_{i+1}) at u_i, $i = 0, \ldots, k-1$. We have $x + 1 \in I_i$ by definition of the path $u_0 = x, u_1, \ldots, u_k = x + 1$ and $u_{i+1} \in I_i$ as (u_i, u_{i+1}) is the unique shortest path from u_i to u_{i+1}. Also, $x \notin I_i$, $i > 0$, by definition of the path $u_0 = x, u_1, \ldots, u_k = x + 1$, and $u_i \notin I_i$, $0 \leq i \leq k - 1$, as the LIRS is strict. In particular, $[x + 1, u_1] \subseteq I_0$, and $x \notin I_0$ implies that $u_1 > x$ as I_0 is linear. More generally, for $i \geq 1$, $[x + 1, u_{i+1}] \subseteq I_i$ and $x \notin I_i$ implies $u_{i+1} > x$ because I_i is linear. Now, $u_i > x$ and $u_i \notin I_i$ implies $u_i > u_{i+1}$. Therefore, $x+1 < u_{i+1} < u_i$ for $0 < i < k - 1$.

Similarly, one can show that $x - 1 > v_{i+1} > v_i$ for $0 < i < m - 1$.

A direct consequence of Lemma 1 is that $d(x, x + 1) \leq n - x$, and $d(x, x - 1) \leq x - 1$, and hence we have:

Corollary 1 *In a network supporting a strict LIRS, $d(i, i + 1) \leq \min\{i, n - i\}$.*

This corollary does not yield any satisfactory upper bound on the message-complexity of the up/down protocol for strict LIRS as it merely induces the trivial upper bound $\sum_{i=1}^{n-1} d(i, i + 1) = O(n^2)$. Lemma 1 is, however, important as it states that, in strict LIRS, once a target node x has received the up copy of the message in the up/down protocol, no node of label smaller or equal to x will be visited anymore, and symmetrically for the down copy. Note that this is not enough to conclude that the message-complexity of the up/down protocol has a message-complexity $O(n)$ as nodes of large labels may be visited a non-constant number of times in the up/down protocol. Nevertheless, Lemma 1 will be extensively used throughout this paper. The following results allow a reader to understand the subtle difference between LIRS and strict LIRS.

Lemma 2 *Let G be a network supporting a LIRS. Let $u_0 = x, u_1, \ldots, u_k = x + 1$ (resp. $v_0 = x, v_1, \ldots, v_m = x - 1$) be the shortest path from node x to node $x + 1$ (resp. to node $x - 1$) set by the LIRS. Then, for every i, $k > i > 1$, we have:*

(i) $u_i > x + 1$,

and

(ii) $u_i < u_j$ *for every j, $1 < j \leq i - 2$.*

(Respectively, for every i, $1 < i < m$: (i) $v_i < x - 1$, and (ii) $v_i > v_j$ for every j, $1 < j \leq i - 2$.)

Proof. Let I_i be the interval of edge (u_i, u_{i+1}) at u_i, $i = 0, \ldots, k-1$. By definition, we have $x+1 \in I_i$ and $u_{i+1} \in I_i$. For $i \geq 1$ we also have $x \notin I_i$ (otherwise there would exist a shorter path from x to $x + 1$). Thus $u_i > x + 1$ for $1 < i < k$ as the intervals are linear. So Property (i) holds.

For proving Property (ii), we note that, if $i \geq 1$, then $u_j \notin I_i$ for every $j \leq i - 1$ by definition of the path $x = u_0, u_1, \ldots, u_k = x + 1$. Therefore, since $u_j > x + 1$ for all j, $1 < j < k$, we have $u_{i+1} < u_j$ for every j, $1 < j \leq i - 1$.

The results for the v_i's are obtained in a similar way.

As a consequence of Lemma 2, we have $d(x, x + 1) \leq n - x + 1$, and $d(x, x - 1) \leq x$, and hence we get a result similar to Corollary 1:

Corollary 2 *In a network supporting an LIRS, $d(i, i + 1) \leq 1 + \min\{i, n - i\}$.*

We now analyze the relation between the labels of nodes in two consecutive sequences of intermediate nodes, $Y_{r-1} = \{u_1, \ldots, u_{\ell_{r-1}}\}$ and $Y_r = \{v_1, \ldots, v_{\ell_r}\}$ say. Let us consider the two following examples:

– Assume $X_r = \{x\}$ and $\ell_r \geq 2$. (Recall that the form of W is $W = \ldots Y_{r-1}, X_r, Y_r \ldots$) If the network supports a strict LIRS, then, by considering the interval on (x, v_1) at x if $v_1 \neq u_{\ell_{r-1}}$ or the interval on (v_1, v_2) at v_1, otherwise, one can easily check that $u_{\ell_{r-1}} > v_2$.
– Assume $X_r = \{x, x+1\}$ and $\ell_r \geq 3$. If the network supports a strict LIRS then one can check similarly that $u_{\ell_{r-1}} > v_3$.

Remark. The two previous examples show that, by "jumping" over some intermediate nodes, one can find a strictly decreasing sequence of intermediate nodes. The problem is however more complex if $X_r = \{x\}$ with $\ell_r = 1$, or if $X_r = \{x, x+1\}$ with $\ell_r \leq 2$. In both cases, there is not enough intermediate nodes to jump in Y_r in order to find an intermediate node of label smaller than $u_{\ell_{r-1}}$. Nevertheless, if $X_r = \{x, x+1\}$, $Y_r = \{v_1, v_2\}$, $X_{r+1} = \{x+2\}$ and if $\ell_{r+1} = |Y_{r+1}|$ is large enough, then one can check that $u_{\ell_{r-1}} > w_2$ where $Y_{r+1} = \{w_1, \ldots, w_{\ell_{r+1}}\}$. However, if ℓ_{r+1} is too small, say $\ell_{r+1} = 1$, then one must consider the next sequence Y_{r+2}, and so on. In fact, as we will see, looking for a node label smaller than $u_{\ell_{r-1}}$ depends on the partial sums

$$\sum_{i=r}^{s}(|Y_i| - |X_i|) = \sum_{i=r}^{s}(\ell_i - k_i)$$

for $s \geq r$. More precisely, this operation depends on the first smallest index s such that this sum is large enough. Roughly speaking, as suggested by the examples above, if $u_{\ell_{r-1}}$ is the last node of Y_{r-1}, then the next node smaller than $u_{\ell_{r-1}}$ may be found in Y_r if $|Y_r| > |X_r|$; otherwise, in Y_{r+1} if $|Y_r| + |Y_{r+1}| > |X_r| + |X_{r+1}|$; and so on. In general, one needs to jump at least as many intermediate nodes as target nodes, in order to make sure to find a node smaller than $u_{\ell_{r-1}}$. The next two lemmas formally state this relationship between consecutive sequences of intermediate nodes. Again, the goal is to answer the following: given an intermediate node $w_i \in W$, where can we find another intermediate node $w_j \in W$, $j > i$, whose label is smaller, or at least not larger, than the label of w_i? (Lemma 1 answers this question if w_i is not the last node of an intermediate sequence in a strict LIRS network, and Lemma 2 answers this question if w_i is neither the last nor the penultimate node of an intermediate sequence in a LIRS network.)

For any set of nodes $S \subseteq V$ and any node $u \in V$, we denote by $d(u, S)$ the distance between u and S, that is $d(u, S) = \min_{v \in S} d(u, v)$. Recall that ϱ is the number of target sequences X_i's.

Lemma 3 *Let G be a network supporting a strict LIRS and let u be a node of G. Let $r \in \{1, \ldots, \varrho\}$ and assume $d(u, X_r) \leq \delta$. Assume that there exists $s \geq r$ such that $\sum_{i=r}^{s}(\ell_i - k_i) > \delta - 1$ and $\sum_{i=r}^{s'}(\ell_i - k_i) \leq \delta - 1$ for every s', $r \leq s' < s$. Let $\kappa = k_s + \sum_{i=r}^{s-1}(k_i - \ell_i) + \delta$ and let $Y_s = v_1, \ldots, v_{\ell_s}$. Assume that $u > x$ for every $x \in \bigcup_{i=r}^{s} X_i$ and assume that $w \geq u$ for every intermediate node $w \in \bigcup_{i=r}^{s-1} Y_i$. Then $u > v_\kappa$.*

We first note that $\sum_{i=r}^{s}(\ell_i - k_i) \geq \delta$ implies that $\kappa \leq \ell_s$. Note also that, if $s > r$, then $\sum_{i=r}^{s-1}(\ell_i - k_i) \leq \delta - 1$ implies that $\kappa \geq k_s + 1 \geq 2$. If $s = r$, then $\kappa = k_s + \delta \geq k_s \geq 1$. Therefore, v_κ is well defined. To prove Lemma 3, we use the following result:

Claim 1 *Under the hypothesis of Lemma 3, $d(u, x_s) \leq \kappa - 1$.*

Proof. If $s = r$ then $d(u, x_s) \leq d(u, X_r) + k_s - 1 \leq \delta + k_s - 1 = \kappa - 1$. Let us show that the same result holds if $s > r$. For that purpose, we show by induction on i that, assuming $s > r$, we have

$$d(u, y_i) \leq (\delta - 1) + \sum_{j=r}^{i}(k_j - \ell_j) \quad \text{for every } i = r, \ldots, s-1. \tag{2}$$

- *Initial step* $i = r$. Let $Y_r = u_1, \ldots, u_{\ell_r}$. Note that $y_r = u_{\ell_r}$. Let I_1 be the interval on the edge (x_r, u_1) at x_r and, for $i > 1$, let I_i be the interval on the edge (u_{i-1}, u_i) at u_{i-1}. For every $i \geq 1$, we have $x_r + 1 \in I_i$, and $x_r \notin I_i$. From Lemma 1, we also have $u_1 > \ldots > u_{\ell_r} > x_r + 1$. By hypothesis of Lemma 3, $u > x$ for every $x \in \bigcup_{i=r}^{s} X_i$, and $u \leq u_{\ell_r}$. Thus, since $u_i \in I_i$, we get that $u \in I_i$ for every $i = 1, \ldots, \ell_r$. Therefore, a shortest path from x_r to u goes through $u_1, \ldots, u_{\ell_r}$. As a consequence $\ell_r + d(y_r, u) \leq k_r + d(u, X_r) - 1$, and thus $d(u, y_r) \leq (\delta - 1) + (k_r - \ell_r)$.
- *Induction step.* Assume that $d(u, y_i) \leq (\delta - 1) + \sum_{j=r}^{i}(k_j - \ell_j)$ for some i, $r \leq i < s - 1$, and let us show that $d(u, y_{i+1}) \leq (\delta - 1) + \sum_{j=r}^{i+1}(k_j - \ell_j)$. For the same reasons as for the case $i = r$, a shortest path from the last node x_{i+1} of X_{i+1} to u goes through all nodes of Y_{i+1}. We get that

$$\begin{aligned} \ell_{i+1} + d(u, y_{i+1}) &\leq d(x_{i+1}, u) \leq k_{i+1} + d(u, y_i) \\ &\leq (\delta - 1) + k_{i+1} + \sum_{j=r}^{i}(k_j - \ell_j). \end{aligned}$$

  And thus $d(u, y_{i+1}) \leq (\delta - 1) + \sum_{j=r}^{i+1}(k_j - \ell_j)$. This completes the proof of (2).

A consequence of (2) is that $d(u, x_s) \leq (\delta - 1) + k_s + \sum_{j=r}^{s-1}(k_j - \ell_j) = \kappa - 1$, which completes the proof of the claim.

Proof of Lemma 3. Let I_1 be the interval on edge (x_s, v_1) at x_s, and, for $i > 1$, let I_i be the interval on edge (v_{i-1}, v_i) at v_{i-1}. We have $x_r + 1 \in I_i$, $v_i \in I_i$, and $x_r \notin I_i$. Therefore, since $u > x + 1$, we have: if $u < v_i$ then $u \in I_i$. Thus if $u < v_i$ for every $i \in \{1, \ldots, \kappa - 1\}$, then $d(u, x_s) > \kappa - 1$, a contradiction with Claim 1. Thus $u \geq v_i$ for some $i \leq \kappa - 1$, and therefore $u > v_\kappa$ by Lemma 1.

The difference between Lemma 1 and Lemma 2 engenders the following adaptation of Lemma 3 for networks supporting non strict LIRS.

Lemma 4 *Let G be a network supporting an LIRS and let u be a node of G. Let $r \in \{1, \ldots, \varrho\}$ and assume that $d(u, X_r) \leq \delta$. Assume that there exists an $s \geq r$ such that $\sum_{i=r}^{s}(\ell_i - k_i) \geq$*

$2(s-r+1)+\delta$ *and* $\sum_{i=r}^{s'}(\ell_i-k_i) < 2(s'-r+1)+\delta$ *for every* s', $r \le s' < s$. *Let* $\kappa = k_s + \sum_{i=r}^{s-1}(k_i-\ell_i) + 2(s-r)+\delta$ *and let* $Y_s = v_1,\ldots,v_{\ell_s}$. *Assume that* $u > x$ *for every* $x \in \bigcup_{i=r+1}^{s} X_i$ *and that* $w > u$ *for every intermediate node* $w \in \bigcup_{i=r}^{s-1}(Y_i \setminus \{z_i\})$. *Then,* $\min\{v_{\kappa+1}, v_{\kappa+2}\} \le u$.

Again, we note that $\sum_{i=r}^{s}(k_i-\ell_i) \le 2(r-s-1)-\delta$ implies $\kappa \le \ell_s - 2$. Moreover, if $s=r$ then $\kappa = k_s+\delta \ge 1$ and otherwise $\sum_{i=r}^{s-1}(k_i-\ell_i) > 2(r-s)-\delta$ implies $\kappa > k_s \ge 1$. Thus κ is well defined.

Proof. We proceed in a similar way to the proof of Lemma 3. We first claim that $d(u,z_s) \le \kappa$. This inequality holds for $s=r$ as $d(u,z_r) \le d(u,X_r)+k_r \le \delta + k_r = \kappa$. The proof for $s>r$ is based on an inductive proof of the following property (analogous to the proof of Claim 1). Assuming $s>r$, we have

$$d(u,y_i) \le \sum_{j=r}^{i}(k_j-\ell_j) + 2(i-r) + (\delta+1) \quad \text{for every } i \in \{r,\ldots,s-1\}. \tag{3}$$

- *Initial step* $i=r$. The shortest path from z_r to u set by the IRS goes through all nodes of Y_r. Thus $\ell_r - 1 + d(y_r,u) \le k_r + d(u,X_r)$, and hence $d(y_r,u) \le (k_r-\ell_r)+(\delta+1)$. Therefore (3) holds for $i=r$.
- *Induction step.* We assume (3) holds for i and show that it holds for $i+1$. Again, the shortest path from z_{i+1} to u set by the IRS goes through all nodes of Y_{i+1}, and thus $\ell_{i+1}-1+d(y_{i+1},u) \le k_{i+1}+1+d(u,y_i)$ and therefore $d(y_{i+1},u) \le \sum_{j=r}^{i+1}(k_j-\ell_j)+(\delta+1)+2(i+1-r)$, and (3) holds for $i+1$.

The claim $d(u,z_s) \le \kappa$ then holds by application of (3) since it yields

$$\begin{aligned} d(z_s,u) &\le k_s + 1 + \sum_{i=r}^{s-1}(k_i-\ell_i) + (\delta+1) + 2(s-r-1) \\ &= \kappa. \end{aligned}$$

Now, since $d(u,z_s) \le \kappa$, there is a node in $\{v_2,\ldots,v_{\kappa+1}\}$ such that $v_i \le u$. Otherwise, the path set by the IRS from z_s to u would go through $v_2,\ldots,v_{\kappa+1}$, and thus would not be a shortest path. Let i be the smallest index in $\{2,\ldots,\kappa+1\}$ such that $v_i \le u$. If $i < \kappa+1$, then $v_{\kappa+2} < u$ from Lemma 2. If $i = \kappa+1$, then $v_{\kappa+1} \le u$ (actually, $v_{\kappa+1} = u$).

3 Strict linear interval routing schemes

In this section, we show that the message-complexity of the up/down protocol is at most $3n$ on a network of order n supporting a strict LIRS.

3.1 Partition: a sequence-decomposition algorithm

Assume first that the source is node 1. We make use of the sequence W as introduced in (1). Since the total number of target nodes is n, we have $\sum_{i=1}^{\varrho}|X_i| = n$. Therefore, the aim of the remainder of the proof is to bound the total number $\sum_{i=0}^{\varrho}|Y_i|$ of intermediate nodes. For that purpose, we will partition the intermediate nodes of W into three types of (pairwise disjoint) subsequences. We call the result of this partition the *sequence-decomposition*. Every intermediate node appears exactly once in this decomposition. The target nodes do not appear in the decomposition. More precisely, the decomposition will be composed of an *active thread*, *dead-end threads* and *jumped threads*. Here the terminology "thread" refers to a sequence on non-necessarily consecutive nodes of W. The active thread is built by a walk starting from w_0 up to $w_{\tau+1}$. Along the construction of the sequence-decomposition, some parts of the active thread become dead-end threads. At the end of the decomposition, the two extremities of the active thread are w_0 and $w_{\tau+1}$ and the total number of nodes in the active thread will be at most $O(n)$. Also, at the end of the construction, the sum of the lengths of the dead-end threads and the sum of the lengths of the jumped threads will be both bounded by $O(n)$. Therefore, the total number of intermediate nodes will be $O(n)$, and $|W| = O(n)$. A careful analysis of the constants will actually show that $|W| \le 3n$.

Let us be more specific. Again, the decomposition is performed by visiting all intermediate nodes of the sequence W from w_0 to $w_{\tau+1}$, and in this way constructing the active thread. One may "jump" over some intermediate nodes if the length of the jump is able to be bounded. The terminology "jump" refers to the remark in Sect. 2. Jumped intermediate nodes form a jumped thread. One may also backtrack along the active thread for a bounded number of nodes. The nodes along which the backtrack occurs form a dead-end thread.

The decomposition requires two parameters: the *mark* m and the *direction* d. The role of the mark is to keep track of the progression along the sequence W. In general, the mark indicates the current position, or, more precisely, the index i of the current set Y_i. The mark is an important parameter because of the backtracks. When a backtrack occurs, the mark is set to hold the current maximum position ever reached in the sequence W. The direction indicates whether or not one is currently backtracking. In the affirmative, $d=-1$ and $d=+1$ otherwise.

Initially, the active thread is reduced to w_0 and there is no dead-end thread nor jumped thread. The mark m is set to 0, and the direction is set to $+1$.

The construction of the sequence-decomposition is precisely described in Algorithm 1. The explanations of the several steps of the construction are given below.

In Case 1, the construction is currently visiting some Y_r. While the last node of this sequence of intermediate nodes is not reached, the active thread is updated by adding all the forthcoming nodes of the current sequence. Informally, from Lemma 1, the size of the active thread will not increase too much since the labels of the nodes are in a strictly decreasing order.

Case 2 happens in particular when the last node of the current sequence Y_r is reached (see Fig. 2a). The definition of s is motivated by Lemma 3. If s does not exist, the construction stops. The jumped thread has then a length bounded by $\sum_{i=r+1}^{\varrho}|X_i|$. In general, every jumped thread b will have a length bounded by $\sum_{i\in\mathcal{I}_b}|X_i|$ for some index set $\mathcal{I}_b$, so that $\mathcal{I}_b \cap \mathcal{I}_{b'} = \emptyset$ if $b' \ne b$. If s does exist, then we make a jump

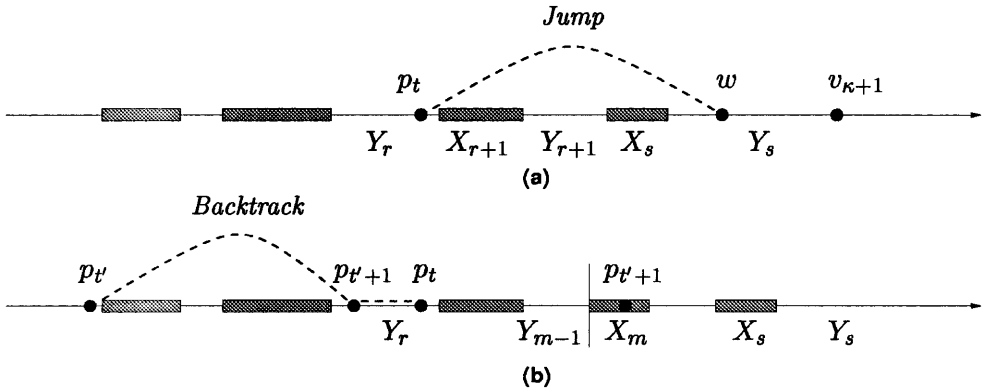

Fig. 2. Construction of the sequence-decomposition

in the sequence W as explained hereafter. (Note that, as in Lemma 3, $2 \leq \kappa \leq \ell_s$.)

In Case 2.1.1, we simply jump at the next intermediate node w whose label is smaller than the label of the current node. Case 2.1.2 can be seen as an extremal case of Case 2.1.1. We know from Lemma 3 that jumping at v_κ is sufficient as the label of that node is smaller than the label of the current node p_t. The mark is updated to contain the index of the sequence of intermediate nodes Y_σ reached after the jump.

Case 2.2 is in fact the most difficult case to explain (see Fig. 2b) primarily because a backtrack occurs. This backtrack is motivated by the fact that one cannot find an intermediate node with a label smaller than that of p_t. Informally, we backtrack along the active thread $p_0, p_1, \ldots, p_t$ until we reach a node $p_{t'}$, $t' < t$, for which Lemma 3 may be applied. Note that t' is well defined since $p_0 \notin \bigcup_{i=1}^{\varrho} X_i = \{1, \ldots, n\}$. We will show that the length of a dead-end thread that results from a backtrack is bounded by $\sum_{i=m+1}^{\sigma} |X_i|$. We will also see that the possible jumped thread set from the value of $p_{t'}$ has a length bounded by $\sum_{i=m+1}^{\sigma-1} |X_i|$. Fig. 2b illustrates the specific case in which the backtrack goes across a former jump.

Now, let us analyze the construction of the sequence-decomposition.

Claim 2 *At any step of the decomposition, if $d = +1$, then $p_t \in Y_m$.*

Proof. Initially this claim holds. Assume the claim holds before some step i and consider the several cases of Algorithm 1. In Case 1, p_{t+1} will still be in Y_m, and d will still be $+1$. In Case 2.1, the direction is set to $+1$ and $p_{t+1} \in Y_m$ by definition of the setting of the mark m in both Cases 2.1.1, and 2.1.2. Case 2.2 sets d to -1. The claim also holds after step i.

Lemma 5 *The construction of the sequence-decomposition given in Algorithm 1 produces a set of threads in which every intermediate node appears exactly once.*

Proof. After every step i, let us define m_i as the resulting mark and

$$q_i = \begin{cases} p_t & \text{if } p_t \in Y_{m_i}; \\ \text{last node of } Y_{m_i} & \text{otherwise.} \end{cases}$$

We claim that, at every step i, all intermediate nodes before q_i appears exactly once in the sequence-decomposition. Initially this claim holds. Assume it holds before step i. In Case 1, from Claim 2, $q_{i-1} = p_t \in Y_{m_{i-1}}$. The active thread is upgraded by adding the next node of the sequence, $Y_{m_i} = Y_{m_{i-1}}$, and thus the claim holds after step i. In Case 2.1, m_i is set such that $p_{t+1} \in Y_{m_i}$ and thus $q_i = p_{t+1}$. Every intermediate node between the last node of $Y_{m_{i-1}}$ (that is q_{i-1}) and p_{t+1} (that is q_i) are put in a jumped thread and so the claim holds after step i. In Case 2.2, some part of the active thread becomes a dead-end thread. By the setting of m_i, all the intermediate nodes not yet assigned to any type of threads, that is all intermediate nodes in $\bigcup_{j=m_{i-1}+1}^{m_i} Y_j$, are put in a jumped thread. Thus every intermediate node between q_{i-1} and q_i is put in a jumped thread. Therefore, the claim holds after step i, thus completing the proof of the claim.

We complete the proof of the lemma by noticing that, after every step, either the active thread is increased, or a part of the active thread, is put in a dead-end thread. By the setting of the mark, an intermediate-node put in a dead-end thread or a jumped thread will not be considered anymore in the further steps. Therefore the construction of the sequence-decomposition of Algorithm 1 ends after a finite number of steps.

3.2 Message-complexity of the up/down protocol

From the sequence-decomposition algorithm, we now compute the message-complexity of the up/down protocol.

Lemma 6 *The number of intermediate nodes in the active thread is at most n, excluding w_0 and $w_{\tau+1}$. More precisely, the labels of the nodes of the active thread, including w_0 and $w_{\tau+1}$, form a decreasing subsequence in $n+1, \ldots, 0$.*

Proof. To prove the lemma, let us again go through the several cases of every step of the decomposition. Let p_t be the last node of the current active thread. In Case 1, Lemma 1 ensures that $p_{t+1} < p_t$. In Case 2, if s does not exist, then the next node of the active thread is $w_{\tau+1} = 0$ which is smaller than every other node of the current active thread. Let us assume that s exists. Since Case 2.2 does not add a new node to the active thread, we focus on case 2.1. In Case 2.1.1, the new node added to the active thread is, by definition, smaller than p_t. In Case 2.1.2, $v_\kappa < p_t$ by application of Lemma 3.

Lemma 7 *The number of nodes in the dead-end threads is at most n.*

Proof. A dead-end thread is composed of a sequence of nodes that were formerly in the active thread and through which

Algorithm 1 One step of the construction of the sequence-decomposition for a strict LIRS

The current active thread is $P = p_0, p_1, \ldots, p_t$, with $p_0 = w_0$. Let r be such that $p_t \in Y_r$, and let $Y_r = u_1, \ldots, u_{\ell_r}$.

Case 1: $p_t = u_i, i < \ell_r$ and $d = +1$. Here, the active thread is updated to $p_0, p_1, \ldots, p_t, u_{i+1}$. There is neither a new dead-end thread nor a new jumped thread. The mark and the direction are not modified.

Case 2: $p_t = u_{\ell_r}$ or $d = -1$. Let $\delta = d(p_t, X_{m+1})$ and $s \geq m+1$ be the smallest index such that $\sum_{i=m+1}^{s}(\ell_i - k_i) > \delta - 1$. If s does not exist, then the active thread is updated to $p_0, p_1, \ldots, p_t, w_{\tau+1}$, there is no new dead-end thread, all intermediate nodes between the last node of Y_m and $w_{\tau+1}$ form a new jumped thread and the construction stops. If s exists, then let $\kappa = \delta + k_s + \sum_{i=m+1}^{s-1}(k_i - \ell_i)$. Assume that $Y_s = v_1, \ldots, v_{\ell_s}$, two cases are considered:

Case 2.1: $p_t \notin \bigcup_{i=m+1}^{s} X_i$. Then again two cases may occur.

Case 2.1.1: There exists $w \in \{v_1, \ldots, v_{\kappa-1}\} \cup (\bigcup_{i=m+1}^{s-1} Y_i)$ such that $w < p_t$.

Then pick the first node w of that type. The active thread is updated to $p_0, p_1, \ldots, p_t, w$, and all intermediate nodes between the last node of Y_m and w form a new jumped thread. Assume $w \in Y_\sigma$, then the mark m is set to σ.

Case 2.1.2: For any $w \in \{v_1, \ldots, v_{\kappa-1}\} \cup (\bigcup_{i=m+1}^{s-1} Y_i)$, $w \geq p_t$.

Then the active thread is updated to $p_0, p_1, \ldots, p_t, v_\kappa$, and all intermediate nodes between the last node of Y_m and v_κ form a new jumped thread. The mark m is set to s.

In both cases, there is no new dead-end thread and the direction d is set to $+1$.

Case 2.2: $p_t \in \bigcup_{i=m+1}^{s} X_i$. Then the direction d is set to -1. Let $t' \in \{0, \ldots, t-1\}$ be the largest index such that $p_{t'} \notin \bigcup_{i=m+1}^{s} X_i$. All nodes $p_{t'+1}, \ldots, p_t$ form a new dead-end thread. Assume $p_{t'+1} \in X_\sigma$, then if $\sigma > m+1$, all intermediate nodes of $\bigcup_{i=m+1}^{\sigma-1} Y_i$ form a new jumped thread. The mark m is updated to $\sigma - 1$.

a backtrack occurred. Every backtrack is driven by a set of target nodes $\bigcup_{i=m+1}^{\sigma} X_i$ where $p_{t'+1} \in X_\sigma$. Let $X_\sigma = x+1, \ldots, x+k$ and $p_{t'+1} = x+i$, $1 \leq i \leq k$. Assume that $p_t \in X_{m'+1}$ for some m'. Let $X_{m'+1} = x'+1, \ldots, x'+k'$, then $p_t = x'+i'$, $1 \leq i' \leq k'$.

From Lemma 6, the number of nodes of the dead-end thread corresponding to $X_{m+1}, \ldots, X_\sigma$ is bounded by $(k' - i') + \sum_{j=m'+2}^{\sigma-1} |X_j| + i$ because the active thread, traversed in the reverse direction, produced a sequence of nodes of increasing labels. Now, m is updated to $\sigma - 1$ after the backtrack, and thus $X_{m+1}, \ldots, X_{\sigma-1}$ will not be considered anymore when counting the number of nodes in the dead-end threads. X_σ may be considered again in another dead-end thread since possibly $p_t \in X_m$, that is $m' = m$ may occur. However, only the $k - i$ last nodes of X_σ will be involved. As a consequence, the total number of nodes in dead-end threads is bounded by $\sum_{i=1}^{\varrho} |X_i| = n$.

Claim 3 *Assume that two jumps J' and J successively occurred at x (that is J' occurred and then, later in the decomposition, a backtrack led back to x and J occurred). Let $y' \in Y_{\sigma'} = u_1, \ldots, u_{\ell_{\sigma'}}$ and $y \in Y_\sigma$ be the respective extremities of J' and J, and assume $y' = u_i$. Let m and δ (resp. m' and δ') be the setting of the mark and the distance when J (resp. J') occurred. Then $\delta \leq (\delta' - 1) + \sum_{j=m'+1}^{\sigma'-1}(k_j - \ell_j) + (k_{\sigma'} - i)$.*

Proof. By definition, $\delta = d(x, X_{m+1})$. We have $y' \in X_{m+1}$ by the setting of m when backtracking through J' since $p_{t'+1} = y'$. Thus

$$\delta \leq d(x, y').$$

Since the jump J' occurred at x we have $x > \hat{x}$ for any $\hat{x} \in \bigcup_{j=m'+1}^{\sigma'} X_j$, and $x \leq \hat{y}$ for any $\hat{y} \in \bigcup_{j=m'+1}^{\sigma'-1} Y_j$. Therefore, by similar arguments to those proving (2) in the proof of claim 1, $d(y_{m'+1}, x) + \ell_{m'+1} \leq k_{m'+1} - 1 + \delta'$ and thus $d(y_{m'+1}, x) \leq (\delta' - 1) + (k_{m'+1} - \ell_{m'+1})$. More generally, again by similar arguments to those proving (2), $d(y_{\sigma'-1}, x) \leq (\delta' - 1) + \sum_{j=m'+1}^{\sigma'-1}(k_j - \ell_j)$. Since $y' = u_i$, still by the same arguments, we have $i + d(y', x) \leq k_{\sigma'} + d(y_{\sigma'-1}, x)$. Therefore,

$$d(x, y') \leq (\delta' - 1) + \sum_{j=m'+1}^{\sigma'-1} (k_j - \ell_j) + (k_{\sigma'} - i).$$

Combining the two inequalities satisfied by $d(x, y')$ completes the proof of the claim.

Lemma 8 *The number of nodes in the jumped threads is at most n.*

Proof. Similarly to the dead-end threads, the jumped threads are characterized by disjoint sets of target nodes of the form $\bigcup_{i=m+1}^{s} X_i$. Let us first consider the jumped threads created by applying Case 2.1 of Algorithm 1. Assume that a jump J occurred between $x \in Y_r$ and $y \in Y_\sigma$, $r < \sigma \leq s$. If $\sigma < s$, then the number of nodes of the jumped thread is at most $\sum_{i=m+1}^{\sigma} \ell_i - 1$, that is at most $(\delta - 1) + \sum_{i=m+1}^{\sigma} |X_i|$ by definition of δ. If $\sigma = s$, then assume that $Y_s = v_1, \ldots, v_{\ell_s}$, $y = v_i$ with $i \leq \kappa = \delta + k_s + \sum_{i=m+1}^{s-1}(k_i - \ell_i)$. Thus, the number of nodes of the corresponding jumped thread is at most $\sum_{i=m+1}^{s-1} \ell_i + \kappa - 1 \leq (\delta - 1) + \sum_{i=m+1}^{s} |X_i|$. So, in any case, the number of nodes of the jumped thread is at most $(\delta - 1) + \sum_{i=m+1}^{\sigma} |X_i|$. If $\delta > 1$, then this setting corresponds to another jump J' that occurred at x previously, say between x and $y' \in Y_{\sigma'} = u_1, \ldots, u_{\ell'}$, $y' = u_i$. Let m' and δ' be the value of the mark and of the distance when J' occurred respectively. From Claim 3, we have

$$\delta \leq (\delta' - 1) + \sum_{j=m'+1}^{\sigma'-1} (k_j - \ell_j) + (k_{\sigma'} - i).$$

The size of the jump at y' is $i - 1 + \sum_{j=m'+1}^{\sigma'-1} \ell_j$. This implies that the number of node for the two jumps is at most $(\delta' - 1) + \sum_{i=m'+1}^{\sigma'} |X_i| + \sum_{i=m+1}^{\sigma} |X_i|$. We repeat the same argument for δ' until we have considered all jumps that successively occurred at x. It yields that the total number of nodes of the set of jumped threads occurring at the same vertex x is at most $\sum_{i=r+1}^{\sigma} |X_i|$ where Y_σ contains the other extremity of the last jump occurred at x.

The analysis is the same for the jumped threads created in Case 2.2 by proceeding as if the jump occurred between $p_{t'}$ and the last node of $Y_{\sigma-1}$ (recall that $p_{t'+1} \in X_\sigma$).

We conclude the proof by noticing that due to the setting of the mark m, no two jumps correspond to the same set of target nodes. Therefore the total number of nodes is at most $\sum_{i=1}^{\varrho} |X_i| = n$.

Combining Lemmas 6, 7 and 8 with Lemma 5 allows to conclude that the total number of intermediate nodes is at most $3n$.

Remark. At the end of the decomposition, let x be the extremity of the active thread when s cannot be defined, that is when $\sum_{i=m+1}^{s-1}(\ell_i - k_i) \leq \delta - 1$ for every s, $m+1 \leq s \leq \varrho$. As $x \in \bigcup_{i=m+1}^{\varrho} X_i$, all nodes of the active thread belong to $\bigcup_{i=m+1}^{\varrho} X_i$ by application of Lemma 6. By the same lemma, the total number of nodes in the active thread is at most $\sum_{i=m+1}^{\varrho} |X_i|$. Since the sets $X_{m+1}, \ldots, X_\varrho$ were not used to bound the total number of dead-end threads, we can conclude that the sum of the number of nodes in the active thread plus the number of nodes in the dead-end threads is at most n.

As a consequence, the total number of nodes in the sequence-decomposition, and the total number of intermediate nodes, is at most $2n$. Therefore, the total number of nodes in the sequence W is at most $3n$ and the message-complexity of the up/down protocol is at most $3n$.

If the source node is not the node labeled 1, then let $\nu > 1$ be the label of the source. From Lemma 1, the message complexity of the copy going upward is at most $3(n-\nu)$ whereas the message complexity of the copy going downward is at most 3ν.

We finally obtain the following theorem.

Theorem 1 *The message-complexity of the up/down broadcast protocol is at most $3n$ in a network of order n supporting a strict LIRS. As a consequence, networks supporting a strict LIRS allows broadcasting with $O(n)$ message-complexity.*

Remark. The complete bipartite graph $K_{2,n-2}$ supports a strict LIRS for which the up/down protocol uses $2n-4$ messages.

4 Linear interval routing schemes

In order to analyze the up/down protocol in networks supporting LIRS, we partition the sequence W in a manner similar to that in the previous section. This decomposition is given in Algorithm 2. Assume first that the source is labeled 1.

Initially, the active thread is reduced to w_0 and there is neither a dead-end thread nor a jumped thread. According to Lemma 2, the active thread will include every other alternate node. A fourth type of thread is introduced: the *auxiliary thread*. Every other alternate node of the active thread is dropped in the auxiliary thread. Initially, the auxiliary thread is empty and the mark m is set to 0.

Let us point out the main differences appearing at every step of the sequence-decomposition. Case 1 is almost the same as Case 1 in Algorithm 1, that is the decomposition uses the current intermediate sequence to construct the active thread. The only modifications are that every other alternate node is dropped in the auxiliary thread, and that one may stop at u_{ℓ_r-1} or u_{ℓ_r}. Therefore, Case 2 considers also the case $p_t = u_{\ell_r-1}$, which, if true, implies that u_{ℓ_r} is put in the auxiliary thread. Case 2 also differs from Case 2 in Algorithm 1 by the definition of both s and κ. These settings are motivated by Lemma 4. Otherwise, the general structure of Algorithm 2 is the same as Algorithm 1.

Algorithm 2 One step of the construction of the sequence-decomposition for an LIRS

The current active thread is $P = p_0, p_1, \ldots, p_t$, with $p_0 = w_0$. Let r be such that $p_t \in Y_r$, and let $Y_r = u_1, \ldots, u_{\ell_r}$.

Case 1: $p_t = u_i, i < \ell_r - 1$ and $d = +1$. Here, the active thread is updated to $p_0, p_1, \ldots, p_t, u_{i+2}$ and node u_{i+1} is put in the auxiliary thread. There is neither a new dead-end thread nor a new jumped thread. The mark and the direction are not modified.

Case 2: $p_t = u_{\ell_r}$ or $p_t = u_{\ell_r-1}$ or $d = -1$. Then let $\delta = d(p_t, X_{m+1})$ and $s \geq m+1$ be the smallest index such that $\sum_{i=m+1}^{s}(\ell_i - k_i) \geq 2(s-m) + \delta$. If s does not exist, then the active thread is updated to $p_0, p_1, \ldots, p_t, w_{\tau+1}$, there is no new dead-end thread, all intermediate nodes between the last node of Y_m and $w_{\tau+1}$ form a new jumped thread and the construction stops. If s exists, then let $\kappa = \delta + k_s + \sum_{i=m+1}^{s-1}(k_i - \ell_i) + 2(s-m-1)$. If $p_t = u_{\ell_r-1}$ and $d = +1$, then u_{ℓ_r} is put in the auxiliary thread. Assume that $Y_s = v_1, \ldots, v_{\ell_s}$, two cases are considered:

Case 2.1: $p_t \notin \bigcup_{i=m+1}^{s} X_i$. Then again two cases may occur.

Case 2.1.1: There exists $w \in \{v_1, \ldots, v_\kappa\} \cup (\bigcup_{i=m+1}^{s-1}(Y_i \setminus \{z_i\}))$ such that $w \leq p_t$.

Pick the first node w of that type. The active thread is updated to $p_0, p_1, \ldots, p_t, w$, and all intermediate nodes between the last node of Y_m and w form a new jumped thread. Assume $w \in Y_\sigma$, the mark m is updated to σ.

Case 2.1.2: For any $w \in \{v_1, \ldots, v_\kappa\} \cup (\bigcup_{i=m+1}^{s-1}(Y_i \setminus \{z_i\}))$, $w > p_t$.

Then the active thread is updated by setting $p_{t+1} = \min\{v_{\kappa+1}, v_{\kappa+2}\}$. All intermediate nodes between the last node of Y_m and p_{t+1} form a new jumped thread. The mark m is set to s.

In both cases, there is no new dead-end thread and the direction d is set to $+1$.

Case 2.2: $p_t \in \bigcup_{i=m+1}^{s} X_i$. The direction d is set to -1. Let $t' \in \{0, \ldots, t-1\}$ be the largest index such that $p_{t'} \notin \bigcup_{i=m+1}^{s} X_i$. All nodes $p_{t'+1}, \ldots, p_t$ form a new dead-end thread. Assume that $p_{t'+1} \in X_\sigma$, then if $\sigma > m+1$, all intermediate nodes of $\bigcup_{i=m+1}^{\sigma-1} Y_i$ form a new jumped thread. The mark m is updated to $\sigma - 1$.

Lemma 9 *The construction of the sequence-decomposition given in Algorithm 2 produces a set of threads in which every intermediate node appears exactly once.*

The proof of this lemma follows exactly the same lines as the proof of Lemma 5. It is therefore omitted. Recall that ϱ denotes the number of sequences X_i in W.

Lemma 10 *The number of intermediate nodes in the active thread is at most $n + \varrho$, excluding w_0 and $w_{\tau+1}$. More precisely, the labels of the nodes of the active thread form a non-*

increasing sequence $p_0, \ldots, p_t$ from $p_0 = n+1$ to $p_t = 0$ such that the number of times that $p_i = p_{i-1}$ is at most ϱ.

Proof. From Lemma 2, Case 1 ensures that $p_{t+1} = u_{i+2} < u_i = p_t$. In Case 2, if s does not exist, then $p_{t+1} = w_{\tau+1} = 0 < p_t$. So assume that s exists. There is no setting of the active thread in Case 2.2. In case 2.1.1, a jump occurs and $p_{t+1} \leq p_t$ by definition. In case 2.1.2, a jump occurs also and $p_{t+1} \leq p_t$ from Lemma 4. The number of jumps is at most ϱ.

Lemma 11 *The number of nodes either in dead-end threads or in the active thread is at most $n + \varrho$.*

Proof. By similar arguments to those in the proof of Lemma 7, and using the same notation, the number of nodes of a dead-end thread corresponding to the sets $X_{m+1}, \ldots, X_\sigma$ is at most $(k' - i') + \sum_{j=m'+1}^{\sigma-1} |X_j| + i$ plus the number of jumps occurring in the portion $p_{t'+1}, \ldots, p_t$ of W. This implies that a total number of nodes in dead-end threads of at most $\sum_{i=1}^{\varrho} |X_i| + \varrho = n + \varrho$.

Now, as observed in the strict LIRS case, one can combine the bound for dead-end threads with the bound for the active thread. Let $x \in \bigcup_{i=m+1}^{\varrho} X_i$ be the last node of the active thread (different from $w_{\tau+1}$). The total number of nodes in the active thread is at most $(k' - i') + \sum_{i=m+2}^{\varrho} |X_i| + j$, where j is the number of jumps of the active thread. The sets $X_{m+2}, \ldots, X_\varrho$ were not used to enumerate the nodes in the dead-end threads and the total number of jumps cannot exceed ϱ. Therefore, the total number of nodes either in the active thread or in dead-end threads is at most $n + \varrho$.

Lemma 12 *The number of nodes in the auxiliary thread is at most $n + \varrho$.*

Proof. This is a direct consequence of Lemma 11 as the number of nodes in the auxiliary thread does not exceed the number of nodes either in the active thread or in dead-end threads. Indeed, at most one node is dropped in the auxiliary thread for every node entering the active thread. Some nodes formerly in the active thread become member of a dead-end thread.

Lemma 13 *The number of nodes in the jumped threads is at most $n + 3\varrho$.*

Proof. Let us first consider a jump created by application of Case 2.1. Let J be a jump corresponding to the sets $X_{m+1}, \ldots, X_s$. Assume that J occurred between $x \in Y_r$ and $y \in Y_\sigma$, $r < \sigma \leq s$.

If $\sigma < s$, then the jump was over at most $\sum_{i=m+1}^{\sigma} \ell_i - 1$ intermediate nodes, with $\sum_{i=m+1}^{\sigma} \ell_i < \sum_{i=m+1}^{\sigma} k_i + 2(\sigma - m) + \delta$ by definition of δ.

If $\sigma = s$, then let $Y_s = v_1, \ldots, v_{\ell_s}$, $y = v_i$. This implies $i \leq \kappa + 2 = \delta + k_s + \sum_{i=m+1}^{s-1} (k_i - \ell_i) + 2(s - m)$.

Therefore, in both cases, the number of jumped intermediate nodes is at most $(\delta - 1) + \sum_{i=m+1}^{\sigma} |X_i| + 2(\sigma - m)$. If $\delta > 2$, then this setting corresponds to another jump J' that occurred at x previously, say between x and $y' \in Y_{\sigma'} = u_1, \ldots, u_{\ell'}$, $y' = u_i$. Let m' and δ' be the value of the mark and the value of the distance, respectively, when this jump occurred. We have

$$\delta \leq (\delta' - 1) + \sum_{j=m'+1}^{\sigma'-1} (k_j - \ell_j) + 2(\sigma' - m') + (k_{\sigma'} - i).$$

Indeed, $\delta = d(x, X_{m+1})$. Moreover, $y' \in X_{m+1}$ by the setting of m when backtracking through J' and as $y' = p_{t'+1}$. Therefore, $\delta \leq d(x, y')$. Conversely, by the same arguments to those of the proof of Claim 3,

$$d(x, y') \leq (\delta' - 1) + \sum_{j=m'+1}^{\sigma'-1} (k_j - \ell_j) + 2(\sigma' - m') + (k_{\sigma'} - i).$$

The size of the jump J' is $i - 1 + \sum_{j=m'+1}^{\sigma'-1} (k_j - \ell_j)$, and thus the total size of the two jumps is at most

$$(\delta' - 1) + \sum_{i=m+1}^{\sigma} |X_i| + 2(\sigma - m) + \sum_{i=m'+1}^{\sigma'} |X_i| + 2(\sigma' - m').$$

If $\delta' > 2$, then one can apply on δ' the same arguments as those that were applied for δ. This implies that the number of nodes that belong to the set of jumped threads occurring at the same vertex $x \in Y_r$ is at most $1 + \sum_{i=r+1}^{\sigma} |X_i| + 2(\sigma - r)$ where Y_σ contains the extremity of the last jump occurred at x.

The analysis is the same for the jumped threads created in Case 2.2, by proceeding as if the jump occurred between $p_{t'}$ and the last node of $Y_{\sigma-1}$ (recall that $p_{t'+1} \in X_\sigma$).

The worst case is reached when every jump is over a single X_i. The cost (in term of number of nodes) of such a jump is $1 + |X_i| + 2 = |X_i| + 3$. This implies a total number of nodes in jumped threads of at most $\sum_{i=1}^{\varrho} |X_i| + 3\varrho$.

The three previous lemmas, together with Lemma 9, show that $\sum_{i=1}^{\varrho-1} |Y_i| \leq 3n + 5\varrho \leq 8n$, and thus $|W| \leq 9n$. If the source node is not the node labeled 1, then let $\nu > 1$ be the label of the source. From Lemma 2, the message complexity of the upward copy is at most $9(n - \nu)$ whereas the message complexity of the downward copy is at most 9ν.

We finally obtain the following theorem.

Theorem 2 *The message-complexity of the up/down broadcast protocol in a network of order n supporting a LIRS is at most $9n$. As a consequence, networks supporting a LIRS allows broadcasting with $O(n)$ message-complexity.*

5 Interval routing schemes

As opposed to LIRS, the labels of the nodes resulting from an IRS play all the same role, meaning that another labeling can be obtained by adding (modulo n) any identical value to all the labels of a given labeling. It is therefore not surprising that results such as Corollary 1 or Corollary 2 do not hold. For instance, let us consider the chain of n nodes (i.e., the graph of node-set $\{x_1, \ldots, x_n\}$ such that there is an edge between nodes of consecutive indices). The nodes of the chain can be labeled as follows: $x_1 = 1$, and $x_i = n - i + 2$ for $i \geq 2$. Let I_i be the interval on (x_i, x_{i+1}) at x_i, $i = 1, \ldots, n-1$, and let J_i be the interval on (x_{i-1}, x_i) at x_i, $i = 2, \ldots, n$. The setting $I_i = [2, n+1-i]$, $J_i = [n-i+3, 1]$, for $i \geq 3$, and $J_2 = [1, 1]$

satisfies the properties of a shortest path IRS. In this setting, $d(1,2) = n-1$, meaning that the difference between any two consecutive labels cannot be bounded. Nevertheless, the labeling of our example satisfies $\sum_{i=1}^{n-1} d(i,i+1) = O(n)$, and therefore the up/down protocol still has a linear message complexity in this case. We will show that this is true for every IRS.

Assume first that the source is labeled 1. Again, in order to analyze the broadcast protocol, we make use of the sequence W defined in (1). For every $x \in \{0,\ldots,n-1\}$, we define

$$L_x : \{1,\ldots,n\} \to \{1,\ldots,n\}$$

such that

$$L_x(u) = 1 + \Big((n+u-x-1) \bmod n\Big).$$

We have $L_x(x) = n$ and $L_x(x+1) = 1$. These "rotations" of the labels allow us to obtain the following lemma.

Lemma 14 *Let G be a network supporting an IRS. Let $u_0 = x, u_1, \ldots, u_k = x+1$ (resp. $v_0 = x, v_1, \ldots, v_m = x-1$) be the shortest path from node labeled x to node labeled $x+1$ (resp. to node labeled $x-1$) set by this IRS.*

- *If the IRS is strict, then $L_x(x) = n > L_x(u_1) > L_x(u_2) > \ldots > L_x(u_{k-1}) > L_x(x+1) = 1$ (resp. $L_{x-1}(x) = 1 < L_{x-1}(v_1) < L_{x-1}(v_2) < \ldots < L_{x-1}(v_{m-1}) < L_{x-1}(x-1) = n$).*
- *Otherwise, for every i, $k > i > 1$, we have $L_x(u_i) > 1$, and $L_x(u_i) < L_x(u_j)$ for every j, $1 < j \leq i-2$ (resp., for every i, $1 < i < m$, $L_{x-1}(v_i) < n$, and $L_{x-1}(v_i) > L_{x-1}(v_j)$ for every j, $1 < j \leq i-2$).*

Proof. The proof is almost identical to the proofs of Lemmas 1 and 2. If the IRS is strict, then let I_i be the interval of edge (u_i, u_{i+1}) at u_i, $i = 0,\ldots,k-1$. For the same reasons as those given in the proof of Lemma 1, we have $x+1 \in I_i$, $u_{i+1} \in I_i$, $x \notin I_i$ and $u_i \notin I_i$. Therefore, $L_x(u_{i+1}) < L_x(u_i)$ for $0 < i < k-1$. Similarly, one can show that $L_{x-1}(v_{i+1}) > L_{x-1}(v_i)$ for $0 < i < m-1$. The proof for arbitrary IRS networks is identical to that of Lemma 2.

Lemma 15 *Let G be a network supporting an IRS and let u be any node of G. Let $r \in \{1,\ldots,\varrho\}$ and assume that $d(u, X_r) \leq \delta$.*

- *If the IRS is strict, then assume that there exists $s \geq r$ such that $\sum_{i=r}^{s}(\ell_i - k_i) > \delta - 1$, and $\sum_{i=r}^{s'}(\ell_i - k_i) \leq \delta - 1$ for every s', $r \leq s' < s$. Let $\kappa = k_s + \sum_{i=r}^{s-1}(k_i - \ell_i) + \delta$ and let $Y_s = v_1,\ldots,v_{\ell_s}$. Assume that, for every $i \in \{r,\ldots,s\}$, and every $x \in X_i$, $L_{x_i}(u) > L_{x_i}(x)$. Moreover, assume that $L_{x_i}(w) \geq L_{x_i}(u)$ for every $i \in \{r,\ldots,s-1\}$ and every intermediate node $w \in Y_i$. Then $L_{x_s}(v_\kappa) < L_{x_s}(u)$.*
- *Otherwise, assume that there exists $s \geq r$ such that $\sum_{i=r}^{s}(\ell_i - k_i) \geq 2(s-r+1) + \delta$, and $\sum_{i=r}^{s'}(\ell_i - k_i) < 2(s'-r+1) + \delta$ for every s', $r \leq s' < s$. Let $\kappa = k_s + \sum_{i=r}^{s-1}(k_i - \ell_i) + 2(s-r) + \delta$, and let $Y_s = v_1,\ldots,v_{\ell_s}$. Assume that, for every $i \in \{r,\ldots,s\}$, and every $x \in X_i$, $L_{x_i}(u) > L_{x_i}(x)$. Moreover, assume that $L_{x_i}(w) > L_{x_i}(u)$ for every $i \in \{r,\ldots,s-1\}$ and every intermediate node $w \in Y_i \setminus \{z_i\}$. Then $\min\{L_{x_s}(v_{\kappa+1}), L_{x_s}(v_{\kappa+2})\} \leq L_{x_s}(u)$.*

Proof. This is similar to the proof of Lemmas 3 and 4, by applying Lemma 14 instead of Lemmas 1 and 2.

The previous results suggest that the sequence-decomposition of Algorithms 1 and 2 may be applied by introducing a relabeling of the L_x's in the appropriate places. More precisely, the sequence decomposition for strict IRS is obtained from the one for strict LIRS by modifying conditions in Cases 2.1.1, and 2.1.2, as follows:

Case 2.1.1 There exists $w \in \{v_1,\ldots,v_{\kappa-1}\} \cup (\bigcup_{i=m+1}^{s-1} Y_i)$, such that: $w \in Y_\sigma$ and $L_{x_\sigma}(w) < L_{x_\sigma}(p_t)$.

Case 2.1.2 For any $w \in \{v_1,\ldots,v_{\kappa-1}\} \cup (\bigcup_{i=m+1}^{s-1} Y_i)$, we have: $w \in Y_\sigma \Rightarrow L_{x_\sigma}(w) \geq L_{x_\sigma}(p_t)$.

Similarly, the sequence decomposition for IRS is obtained from the one for LIRS by modifying conditions in Cases 2.1.1, and 2.1.2, as follows:

Case 2.1.1 There exists $w \in \{v_1,\ldots,v_\kappa\} \cup (\bigcup_{i=m+1}^{s-1}(Y_i \setminus \{z_i\}))$ such that: $w \in Y_\sigma$ and $L_{x_\sigma}(w) \leq L_{x_\sigma}(p_t)$.

Case 2.1.2 For any $w \in \{v_1,\ldots,v_\kappa\} \cup (\bigcup_{i=m+1}^{s-1}(Y_i \setminus \{z_i\}))$, we have: $w \in Y_\sigma \Rightarrow L_{x_\sigma}(w) > L_{x_\sigma}(p_t)$.

The resulting algorithms satisfy the same properties as Algorithms 1 and 2, respectively. There is only one difference: when a jump takes place in an IRS decomposition, say between $p_t \in Y_r$ and $w \in Y_\sigma$, then, by Lemma 15, $L_{x_\sigma}(w) < L_{x_\sigma}(p_t)$ in the strict IRS decomposition and $L_{x_\sigma}(w) \leq L_{x_\sigma}(p_t)$ in the IRS decomposition. This means that, by using rotations of L_x's, the labels of the intermediate nodes between two consecutive targets may be arranged to be in decreasing order. However, we need to check whether this decreasing order is preserved between different sequences of intermediate nodes, that is if we apply different rotations. The next lemma is the principle of the generalization to IRS networks of the results obtained for LIRS networks.

Lemma 16 *Let p_i and p_{i+1} be two consecutive nodes of the active thread of the IRS sequence decomposition. Assume that $p_i \in Y_r$ and $p_{i+1} \in Y_\sigma$, $r < \sigma$. Then $L_{x_\sigma}(p_{i+1}) < L_{x_r}(p_i)$ in strict IRS decomposition and $L_{x_\sigma}(p_{i+1}) \leq L_{x_r}(p_i)$, where the maximum number of equalities is at most ϱ, in IRS decomposition.*

Proof. According to the statement of the lemma, there is a jump at p_i and thus we are considering Case 2.1 of the sequence decomposition. Thus $p_i \notin \bigcup_{j=m+1}^{\sigma} X_j$. More precisely, $p_i \notin \bigcup_{j=r+1}^{\sigma} X_j$. Indeed, if $m \neq r$, then this property holds from the setting of the mark after a backtrack ending at p_i. (Note that several backtracks may lead back to p_i.)

Let us first consider the strict IRS decomposition. From Lemma 15, we have

$$L_{x_\sigma}(p_{i+1}) < L_{x_\sigma}(p_i).$$

Moreover, from Lemma 14, we have $L_{x_r}(p_i) > L_{x_r}(x_r+1)$. If $L_{x_r}(x_\sigma) > L_{x_r}(p_i)$, then we would have that p_i is equal to a target node in $\bigcup_{j=r+1}^{\sigma} X_j$, a contradiction. Therefore $L_{x_r}(x_\sigma) \leq L_{x_r}(p_i)$. Actually, $L_{x_r}(x_\sigma) < L_{x_r}(p_i)$ because, since $p_i \notin X_\sigma$, $x_\sigma \neq p_i$. Thus

$$L_{x_\sigma}(p_i) < L_{x_r}(p_i).$$

By combining the two inequalities on $L_{x_\sigma}(p_i)$, we get $L_{x_\sigma}(p_{i+1}) < L_{x_r}(p_i)$.

The same type of arguments allow to show that $L_{x_\sigma}(p_{i+1}) \leq L_{x_r}(p_i)$ in the sequence-decomposition for IRS.

From the previous result, we get that if $f(i)$ denotes the index such that $p_i \in Y_{f(i)}$, then the active thread $p_0, p_1, \ldots, p_t$, p_{t+1} resulting from the sequence decomposition for strict IRS satisfies $p_0 = n+1$, $p_{t+1} = 0$ and

$$L_{x_{f(i)}}(p_i) > L_{x_{f(j)}}(p_j) \tag{4}$$

for any pair (i,j), $1 \leq i < j \leq t$. There might be up to ϱ equalities in the sequence decomposition for IRS. In other words, the number of nodes in the active thread of the sequence decomposition is n for strict IRS networks and $n+\varrho$ for arbitrary IRS networks. Bounding the number of nodes in the jumped threads, the dead-end threads, and the auxiliary thread, may then be achieved by exactly the same arguments as in Sections 3 and 4. Finally, (4) holds even if the source node is not labeled 1. Therefore, we have:

Theorem 3 *The message-complexity of the up/down broadcast protocol is at most $6n$ in a network of order n supporting a strict IRS and at most $18n$ in a network of order n supporting an IRS. As a consequence, networks supporting a shortest path Interval Routing Scheme allows broadcasting with $O(n)$ message-complexity.*

Corollary 3 *In a network supporting a shortest path Interval Routing Scheme, the average distance between two nodes labeled by two consecutive integers is bounded by a constant.*

Remark. The inherent serial behavior of the up/down protocol (by forwarding an increasing token and a decreasing token) imposes $\Omega(n)$ hops in any network. We could however introduce a variant of the up/down protocol to reduce the number of hops by using more copies of the message. For instance, a new protocol could be obtained by modifying the up/down protocol as follows (we only consider the changes regarding the up copy, the modification for the down copy may be obtained similarly). Each header now contains an interval $[a,b]$ of consecutive targets to reach. The minimum label a represents the current target to reach. Initially, the originator ν sets a to $\nu+1$ and b to n, and sends the up copy toward $\nu+1$. Any intermediate node x which receives a message with header $[a,b]$ proceeds as follows. If $x \notin [a,b]$, then x simply forwards the message toward a. If $x \in [a,b]$ then x creates two copies of the message, one which is sent toward $x+1$ with the header $[x+1,b]$, and another which is sent toward a with header $[a,x-1]$. A message of header $[x,x]$ reaching node x is removed from the network. The asymptotic message-complexity of this variant of the up/down protocol is not higher than the asymptotic message-complexity of the original version (i.e., at most $\sum_{i=1}^{n-1} d(i,i+1)$), but the number of hops of this variant can be significantly smaller than the number of hops of the original version in some networks.

6 Related problems

The fact that n-node networks supporting shortest path interval routing schemes allow broadcasting with $O(n)$ message-complexity can be directly used to solve other problems, such as *leader election* or *distributed spanning tree*. For simplicity, we present only straightforward yet asymptotically optimal solutions here. Far more elegant solutions could be devised, although they must also use $O(n)$ messages.

Recall that, informally, the leader-election is the problem of moving the network from an initial situation where the nodes are in the same computational state, to a final situation where exactly one node is in a distinguished computational state (called *leader*) and all others are in the same state (called *defeated*). The election process may be independently started by any subset of the nodes, called awakened-nodes (any other node is said to be "asleep"). Every node x has a distinct input value $I(x)$ chosen from some infinite totally ordered set and each processor is only aware of its own input value. Similar to most of the solutions commonly used to solve the leader election problem, our strategy elects the node with the lowest input value.

Theorem 4 *Networks supporting a shortest-path interval routing scheme allow leader-election with $O(n)$ message-complexity.*

Proof. Our protocol is a 3-phase protocol:

1. Wake-up node labeled 1 via a down protocol;
2. Identify the node with the lowest input value via an up protocol;
3. Broadcast the name of this node from node n.

More precisely, every awaken node x starts by sending a message (W,x) to node $x-1$ where"W" stands for *wake up* and x is the label of the node in the IRS. Every intermediate node executes the broadcast protocol "down" on messages of type W. Every target node x receiving the message $(W,x+1)$ awakes, if not yet awakened. An awakened node x which has already sent a message (W,x) to $x-1$ does not forward the message $(W,x+1)$, but removes it from the network. A node waking up proceeds as specified before, i.e., it sends a message (W,x) to node $x-1$. When node 1 receives the message $(W,2)$ as a target node, it awakes (if not yet awakened). This completes Phase 1.

Once awakened, node 1 starts Phase 2, and initiates an "up" protocol with message $(E,I(1),1)$ where "E" stands for *elect*. Every node executes protocol "up" on messages of type E with the following modification. When a target node x receives a message (E,I,y), it proceeds as follows: if $I(x) < I$, then x replaces the message (E,I,y) by $(E,I(x),x)$. When n receives the message (E,I,x) as a target node, it elects x as leader if $I < I(n)$, or itself otherwise. Phase 2 is completed.

Phase 3 consists of a broadcast from node n of the identity of the leader selected by node n.

Phase 1 does not generate more than $O(n)$ messages; the number of messages generated by Phase 1 is $\sum_{i=1}^{n-1} d(i,i+1)$ as an awakened node x sends (W,x) to $x-1$ only once. Therefore, from Corollary 3, Phase 1 generates $O(n)$ messages. Phase 2 does not generate more than $O(n)$ messages as its message-complexity is equal to the message-complexity of a broadcast from node 1. Phase 3 is a broadcast, and thus it does not generate more than $O(n)$ messages. Therefore, the message-complexity of the whole protocol is $O(n)$

In the distributed spanning tree problem, every node of a network $G=(V,E)$ must select some of its neighbors (at

least one) such that the graph $T = (V, E')$ where E' is the set of edges linking every node with its selected neighbors, is a tree spanning G.

Theorem 5 *Networks supporting a shortest-path interval routing scheme allow distributed spanning tree to be solved with $O(n)$ message-complexity.*

Proof. As in the leader election protocol, a preliminary phase is performed to wake up the node labeled 1. Once awake, Node labeled 1 broadcasts a message "construct". Upon reception from u of the message "construct", a (non necessarily target) node v proceeds as follows:

- If it is the first time that v received the message "construct", then
  1. v selects u as "parent", and
  2. v sends a message "parent" to u to let u know that v selected it as its parent;
- v executes the instructions of protocol "up" applied to the message "construct".

Once the protocol "up" completes at node n, node n initiates a protocol "down" to broadcast a message "end" to inform that the tree-construction is completed. Once node 1 receives the message "end" as a target node, the whole process is completed. Every node defines its neighbors in the tree by selecting (1) its parent, and (2) all the nodes from which it receives a "parent" message (if any). This protocol has $O(n)$ message-complexity: $O(n)$ messages "construct", $O(n)$ messages "end", and $n-1$ messages "parent".

7 Conclusion

We have identified three directions in which the results presented in this paper could be extended.

1. One may consider the case in which there are two or more intervals at each extremity of every edge. A network supports a k-IRS, if it supports a shortest-path interval routing scheme with at most k intervals at each extremity of every edge.
2. One may consider the case in which the shortest path constraint is relaxed. The *stretch factor* is the maximum ratio between the length of the route set by the routing between two nodes and the distance between these two nodes.
3. One may consider the case of weighted networks, in which every edge has a weight associated to it and the distance between nodes is computed according to these weights.

Let us illustrate the difficulty of these problems by revisiting Lemma 1 in each case.

Let G be a network supporting a strict p-LIRS. Let $u_0 = x, u_1, \ldots, u_k = x+1$ be the shortest path from node labeled x to node labeled $x+1$ set by this routing. Let $I_i^j, j \in \{1, \ldots, p\}$, be the jth interval of the edge (u_i, u_{i+1}) at u_i. There exists j such that $x+1 \in I_i^j$. Also, there exists j' such that $u_{i+1} \in I_i^{j'}$. The fact that $j' \neq j$ could possibly occur makes it difficult to rank the u_i's. The same techniques as those developed in this paper may possibly succeed for small p, but it would be up to the price of a laborious case-analysis.

Let G be a network supporting a strict LIRS with stretch factor s. Let $u_0 = x, u_1, \ldots, u_k = x+1$ be the path from node labeled x to node labeled $x+1$ set by this routing. Let I_i be the interval of the edge (u_i, u_{i+1}) at u_i. We have $x+1 \in I_i$. However, there is no evidence that $u_{i+1} \in I_i$ as the routing from u_i to u_{i+1} may go through a path of length at most s. This makes it difficult to generalize our approach to stretches $s \geq 2$.

Finally, let G be an edge-weighted network supporting a strict LIRS. Let $u_0 = x, u_1, \ldots, u_k = x+1$ be the path from node labeled x to node labeled $x+1$ set by this routing. Let I_i be the interval of the edge (u_i, u_{i+1}) at u_i. We have $x+1 \in I_i$. Again, there is no evidence that $u_{i+1} \in I_i$ as the routing from u_i to u_{i+1} may go through a path whose weighted sum equals the weight of the edge (u_i, u_{i+1}).

In conclusion, our technique does not seem to generalize easily. More powerful tools and new protocols seem to be required to investigate these three problems. As a consequence, we leave these possible extensions as open problems.

As a last remark, we point out that our protocols are very sensitive to faults. For instance, if node 2 is faulty, then broadcasting from node 1 using Protocol up/down would completely fail. We believe that the design of fault-tolerant routing and/or broadcasting protocols based on standard IRS is a challenge that would be worthy of addressing in the future.

Acknowledgements. The authors are thankful to the anonymous referees for their comments which allow significant improvements of the presentation.

References

1. B. Awerbuch, I. Cidon, S. Kutten, Y. Mansour, D. Peleg. Optimal broadcast with partial knowledge. *SIAM J Comput*, 28(2):511–524, 1998
2. B. Awerbuch, O. Goldreich, D. Peleg, R. Vainish. A tradeoff between information and communication in broadcast protocols. *Journal of the ACM*, 37(2):238–256, 1990
3. A. Bar-Noy, S. Guha, J. Naor, B. Schieber. Multicasting in heterogeneous networks. In: *30th ACM Symposium on Theory of Computing (STOC '98)*, pp 448–453, 1998
4. P. de la Torre, L. Naranayan, D. Peleg. Thy neighbor's interval is greener: A proposal for exploiting interval routing schemes. In: *5th International Colloquium on Structural Information and Communication Complexity (SIROCCO '98)*, pp 214–228. Carleton Scientific, 1998
5. K. Diks, E. Kranakis, D. Krizanc, A. Pelc. The impact of knowledge on broadcasting time in radio networks. In: *7^{th} Annual European Symposium on Algorithms (ESA)*, Lectures Notes in Computer Science, Vol 1643 , pp 41–52. Berlin Heidelberg New York: Springer 1999
6. T. Eilam, D. Peleg, R. Tan, S. Zaks. Broadcast in linear messages in IRS representing all shortest paths. Manuscript, 1997
7. M. Flammini, J. van Leeuwen, A. Marchetti-Spaccamela. The complexity of interval routing on random graphs. In: *20^{th} International Symposium on Mathematical Foundations of Computer Sciences (MFCS)*, Lecture Notes in Computer Science, Vol 969, pp 37–49. Berlin Heidelberg New York: Springer 1995
8. P. Flocchini, B. Mans, N. Santoro. Sense of direction: Definitions, properties and classes. *Networks*, 32:165–180, 1998
9. P. Flocchini, B. Mans, N. Santoro. On the impact of Sense of Direction on Message Complexity. *Inform Process Letters*, 63(1):23–31, 1997

10. P. Flocchini, B. Mans, N. Santoro. Sense of direction in distributed computing. In: *12th International Symposium on Distributed Computing (DISC)*, Lecture Notes in Computer Science, Vol 1499, pp 1–15. Berlin Heidelberg New York: Springer 1998
11. P. Fraigniaud, C. Gavoille. Interval routing schemes. *Algorithmica*, 21:155–182, 1998
12. P. Fraigniaud, E. Lazard, Methods and problems of communication in usual networks. *Discrete Applied Mathematics* 53:9–133, 1994
13. G. Frederickson. Searching among intervals and compact routing tables. In: 20^{th} *International Colloquium on Automata, Languages and Programming (ICALP)*, Lecture Notes in Computer Science, Vol 700, pp 28–39. Berlin Heidelberg New York: Springer 1993
14. G. Frederickson, R. Janardan. Seperator-based strategies for efficient message routing. In: 27^{th} *Symposium on Foundations of Computer Science (FOCS)*, pp 428–437, 1986
15. G. Frederickson, R. Janardan. Designing networks with compact routing tables. *Algorithmica*, 3:171–190, 1988
16. G. Gallager, P. Humblet, P. Spira. A distributed algorithm for minimal spanning tree. *ACM Trans Program Lang Syst*, 5(1):66–77, 1983
17. C. Gavoille. A survey on interval routing. *Theor Comput Sci* 245(2):217–253, 2000
18. C. Gavoille, E. Guévremont. Worst case bounds for shortest path interval routing. *J Algorithms*, 27:1–25, 1998
19. C. Gavoille, N. Hanusse. Compact routing tables for graphs of bounded genus. In: 26^{th} *International Colloquium on Automata, Languages and Programming (ICALP)*, Lecture Notes in Computer Science, Vol 1644, pp 351–360. Berlin Heidelberg New York: Springer 1999
20. C. Gavoille, D. Peleg. The compactness of interval routing. *SIAM J Discrete Math*, 12(4):459–473, 1999
21. C. Gavoille, S. Pérennès. Memory requirement for routing in distributed networks. In: 15^{th} *Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pp 125–133, 1996
22. S.M. Hedetniemi, S.T. Hedetniemi, A. Liestman. A survey of gossiping and broadcasting in communication networks, *Networks* 18:319–349, 1986
23. E. Kranakis, D. Krizanc. Lower bounds for compact routing. In: 13^{th} *Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, Lecture Notes in Computer Science, Vol 1046, pp 529–540. Berlin Heidelberg New York: Springer 1996
24. E. Kranakis, D. Krizanc, S. Ravi. On multi-label linear interval routing schemes. In: 19^{th} *International Workshop on Graph Theoretic Concepts in Computer Science (WG)*, Lecture Notes in Computer Science, Vol 790, pp 338–349. Berlin Heidelberg New York: Springer 1993
25. E. Korach, S. Kutten, S. Moran. A modular technique for the design of efficient distributed leader finding algorithms, *ACM Trans on Program Lang Syst*, 12(1):84–101, 1990
26. D. Peleg, E. Upfal. A trade-off between space and efficiency for routing tables. *J ACM*, 36(3):510–530, 1989
27. N. Santoro, R. Khatib. Labelling and implicit routing in networks. *The Computer Journal*, 28(1):5–8, 1985
28. J. van Leeuwen, R. Tan. Interval routing. *The Computer Journal*, 30(4):298–307, 1987