Changing the Routing Protocol without Transient Loops for a Single Destination

Nancy El Rachkidy[a], Alexandre Guitton[a,*]

[a] *Clermont Université, Université Blaise Pascal, LIMOS, BP 10448, F-63000 Clermont-Ferrand, France*
CNRS, UMR 6158, LIMOS, F-63173 Aubière, France

Abstract

Computer networks generally operate using a single routing protocol. However, there are situations where the routing protocol has to be changed (*e.g.*, because an update of the routing protocol is available, or because an external event has triggered a change in the traffic type with different quality of service requirements). In this paper, we show that an uncontrolled transition of the routing protocol might yield to routing loops (even if the involved routing protocols are loop-free). We show that it is possible to achieve a loop-free transition for a single destination, using a mechanism based on several steps. However, reducing the number of steps is NP-hard. We propose an $\mathcal{O}(n^4)$ centralized heuristic (for a network of n nodes), to achieve the transition in few steps in a large variety of scenarios. We also propose a distributed protocol to achieve the same goal with a limited overhead.

Keywords: Routing protocols, routing protocol transition, transient routing loops.

1. Introduction

Computer networks generally operate a single routing protocol which determines which route packets have to follow in order to reach a destination. However, some situations require to change the current routing protocol. For example, this change might be triggered by the availability of a major update of the protocol (or the correction of a security issue). Another example concerns monitoring applications in wireless sensor networks, where the detection of a critical event might trigger the transition from an energy-efficient routing protocol to a delay-sensitive routing protocol [1]. Another last example focuses on the changes in routing decisions caused by major modifications of the topology (due to link or node failures, or due to significant changes in routing metrics) [2, 3, 4].

If nodes are accurately synchronized, they can perform the transition simultaneously from the current routing protocol $\mathcal{R}_1$ to the new routing protocol $\mathcal{R}_2$. However, this solution is often difficult to implement in practice. Indeed, the cost of an accurate synchronization might be prohibitive, or nodes might be operated by different network administrators, leading to different plannings for the transition. We assume in the following that nodes cannot be synchronized in such a precise manner.

If nodes are not synchronized and if nodes perform the transition arbitrarily, transient routing loops might occur, even if the routing protocols are loop-free when considered independently. Figure 1 shows such an example. Initially, all packets towards d are routed according to routing protocol $\mathcal{R}_1$, and $\mathcal{R}_1$ is loop-free (see Figure 1(a)). If nodes are arbitrarily requested to perform the transition to another protocol $\mathcal{R}_2$ (shown on Figure 1(b)), it is possible that c performs the transition first, resulting into the routing depicted on Figure 1(c). In this case, a routing loop occurs between nodes b and c.

Routing loops reduce network performance as they can cause node inaccessibility issues or overload the network. Even if the routing loops caused by the transitions are transient (because all nodes will eventually

*Corresponding author.
Email Address: alexandre.guitton@univ-bpclermont.fr (Alexandre Guitton)
Phone: +33 473177039, Fax: +33 473407639

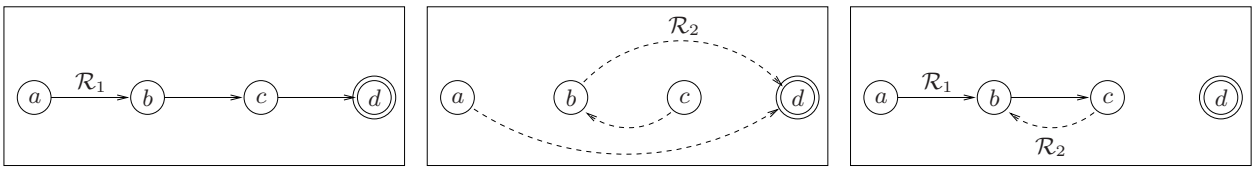


Figure 1: Uncontrolled transitions might yield to routing loops. (a) The initial routing protocol $\mathcal{R}_1$ is loop-free for destination d. (b) The new routing protocol $\mathcal{R}_2$ is also loop-free. (c) A loop occurs between nodes b and c, if a and b route according to $\mathcal{R}_1$ and c routes according to $\mathcal{R}_2$, a loop occurs between nodes b and c.

perform the transition to the new, loop-free routing protocol), our aim is to completely avoid them, as they might have a significant impact for the applications.

In this paper, we show that the transition from a routing protocol $\mathcal{R}_1$ to another routing protocol $\mathcal{R}_2$ can be performed in sequential steps, where all nodes of the same step can perform the transition arbitrarily without causing loops. We propose two mechanisms to perform the transition. They both assume that each routing protocol is loop-free, computes routes for a single destination, and that the routing decisions are static (that is, the next-hop of a node for a given protocol remains the same during the transition). Our two mechanisms aim to reduce the overall number of steps, in order to reduce the transition duration. Our first mechanism is a centralized heuristic based on the computation of strongly connected components. Our second mechanism is a distributed protocol based on neighbor monitoring.

The remainder of the paper is organized as follows. Section 2 formalizes the loop-free transition problem, and shows that reducing the number of steps of a transition is an NP-hard problem. Then, it presents our two mechanisms: a centralized heuristic based on a global knowledge of the topology, and a distributed protocol based on a local knowledge of the state of neighbors. It also starts a brief discussion on routing protocols for several destinations. Section 3 evaluates the performance of our two mechanisms for a several routing protocols. Section 4 describes relevant research works of the literature. Finally, Section 5 concludes this work and gives perspectives.

2. Proposed mechanisms for loop-free transitions

In this section, we present two mechanisms for loop-free transitions. Subsection 2.1 formalizes the problem of loop-free routing protocol transition, as well as the requirements on the steps to perform the transition. Subsection 2.2 shows that building a large first step is NP-hard. Subsection 2.3 presents our centralized heuristic based on strongly connected components. Subsection 2.4 presents our distributed protocol based on neighbor monitoring. Finally, Subsection 2.5 discusses briefly the transition of routing protocols for several destinations.

2.1. Problem formulation

Let V be a set of nodes and $d \in V$ a destination. Let us consider two routing protocols for destination d: $\mathcal{R}_1$ is the routing protocol initially used by nodes, and $\mathcal{R}_2$ is the new protocol to use. For all $i \in \{1, 2\}$ and for all $n \in V$, we denote by $\mathcal{R}_i(n) \in V$ the next-hop of n towards d according to $\mathcal{R}_i$. Given a partition (V_1, V_2) of V (*i.e.*, $V_1 \cup V_2 = V$ and $V_1 \cap V_2 = \emptyset$), we denote by $\mathcal{R}_{V_1,V_2}$ the routing protocol defined in the following way: for all $n \in V_1$, $\mathcal{R}_{V_1,V_2}(n) = \mathcal{R}_1(n)$, and for all $n \in V_2$, $\mathcal{R}_{V_1,V_2}(n) = \mathcal{R}_2(n)$. In other words, nodes of V_1 route according to $\mathcal{R}_1$, and nodes of V_2 route according to $\mathcal{R}_2$. The set of routing decisions of $\mathcal{R}_1$ and $\mathcal{R}_2$ form the set E of the edges of the graph $G = (V, E)$.

Definition 1 (Loop-free step)**.** *Let (V_1, V_2) be a partition of nodes, and $S \subset V_1$ a set of nodes called step. Step S is said to be loop-free if and only if for all $S' \subset S$, the routing protocol $\mathcal{R}_{V_1 \setminus S', V_2 \cup S'}$ is loop-free.*

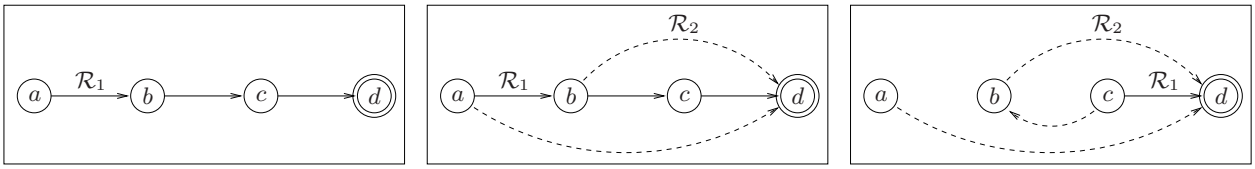
Definition 1 states that a step S is loop-free if and only if all the possible intermediate sub-steps $S' \subset S$ correspond to a loop-free routing protocol. In this way, we can guarantee that the nodes of S can perform the transition of their routing protocol arbitrarily without causing loops.

Definition 2 (Loop-free sequence)**.** *Let $\mathbb{S} = (S_1, \ldots, S_m)$ be a sequence of steps. Sequence $\mathbb{S}$ is said to be loop-free if and only if both conditions apply:*

- $S_1 \cup \ldots \cup S_m = V$,
- *for all $i \in [1; m]$, S_i is a loop-free step on partition (V_1^i, V_2^i), with $V_2^i = S_1 \cup \ldots \cup S_{i-1}$ and $V_1^i = V \backslash V_2^i$.*

Definition 2 states that a sequence $\mathbb{S} = (S_1, \ldots, S_m)$ is loop-free if and only if each step S_i is loop-free, and after step S_m, all nodes belong to $V_2^m \cup S_m = V$ (that is, they have performed the transition to the target routing protocol $\mathcal{R}_2$). Note that the set of nodes running $\mathcal{R}_2$, which is V_2^i, increases as the sequence progresses.

Figure 2 shows an example of loop-free sequence $\mathbb{S} = (\{a, b, d\}, \{c\})$. Figure 2(a) shows the initial routing protocol $\mathcal{R}_1$. Figure 2(b) shows both $\mathcal{R}_1$ and $\mathcal{R}_2$ for nodes of the first step $\{a, b, d\}$, to indicate that these nodes might route according to $\mathcal{R}_1$ (if they have not performed the transition to the routing protocol yet) or $\mathcal{R}_2$ (if they have already perform the transition of the routing protocol). It can be seen that for any routing protocol used by the nodes of the first step, no loop occurs. Figure 2(c) shows $\mathcal{R}_2$ for nodes that have performed the transition of their routing protocol on the previous step, and shows both $\mathcal{R}_1$ and $\mathcal{R}_2$ for the node of the second step. On a side note, it can be noticed that $\mathbb{S}$ is a loop-free sequence with a minimum number of steps, as it is not possible to have both b and c in the same loop-free step.



(a) (b) (c)

Figure 2: An example of loop-free sequence $\mathbb{S} = (\{a, b, d\}, \{c\})$. (a) Initially, all nodes route according to $\mathcal{R}_1$. (b) During the first step, nodes from $\{a, b, d\}$ might route according to $\mathcal{R}_1$ or $\mathcal{R}_2$ without causing loops. (c) During the second step, node c routes according to $\mathcal{R}_1$ or $\mathcal{R}_2$, while all nodes from $\{a, b, d\}$ route according to $\mathcal{R}_2$. At the end of the second step, all nodes route according to $\mathcal{R}_2$.

Theorem 1. *Let G, V_1, V_2, $\mathcal{R}_1$ and $\mathcal{R}_2$ be defined as previously. Let $G' = (V, E')$ with $E' \subset E$, such that for all $x \in V_1$, $(x, \mathcal{R}_1(x)) \in E'$ and $(x, \mathcal{R}_2(x)) \in E'$, and for all $x \in V_2$, $(x, \mathcal{R}_2(x)) \in E'$. Let $\mathcal{C} = \{\mathcal{C}_i\}_i$ be the set of all strongly connected components of G'. For all i, let us denote by $\mathcal{C}'_i \subset \mathcal{C}_i$ a set of nodes that verifies the following properties:*

- $\mathcal{C}'_i \subset V_1$,
- *$G'_i = (V, E'_i)$ does not contain any loop, with E'_i defined as follows:*
  - *if $x \in \mathcal{C}'_i$ then $(x, \mathcal{R}_1(x)) \in E'_i$ and $(x, \mathcal{R}_2(x)) \in E'_i$,*
  - *if $x \in V_1 \backslash \mathcal{C}'_i$ then $(x, \mathcal{R}_1(x)) \in E'_i$,*
  - *if $x \in V_2$ then $(x, \mathcal{R}_2(x)) \in E'_i$.*

Then, $S = \cup_i \mathcal{C}'_i$ is a valid step.

Proof. Let us assume that $\{\mathcal{C}'_i\}_i$ verifies the properties. We have to show that $S = \cup_i \mathcal{C}'_i$ is a valid step, that is, for every $S' \subset S$, $\mathcal{R}_{V_1 \backslash S', V_2 \cup S'}$ is loop-free and leads to destination d. Let S' be an arbitrary subset of S, and let us build $\mathcal{R}_{V_1 \backslash S', V_2 \cup S'}$.

Let us first show that $\mathcal{R}_{V_1 \backslash S', V_2 \cup S'}$ is loop-free. By contradiction, let us suppose that there is a loop $(x_0, x_1, \ldots, x_n)$ in $\mathcal{R}_{V_1 \backslash S', V_2 \cup S'}$, with $x_n = x_0$.

- Suppose here that this loop spans a single strongly connected component $\mathcal{C}_i$. For all k, we have $(x_k, x_{k+1}) \in E_{S'}$. (i) If $x_k \in S'$, then $x_{k+1} = \mathcal{R}_2(x_k)$ by construction of $E_{S'}$ (since $S' \subset S \subset \mathcal{C}'_i$). Because of the properties of $\mathcal{C}'_i$, we have $(x_k, \mathcal{R}_2(x_k)) \in E'_i$. (ii) If $x_k \notin S'$ and $x_k \in V_1$, then $x_{k+1} = \mathcal{R}_1(x_k)$ by construction of $E_{S'}$. We have to consider the two following sub-cases: $x_k \in \mathcal{C}'_i$ and $x_k \notin \mathcal{C}'_i$. If $x_k \in \mathcal{C}'_i$, then $(x_k, \mathcal{R}_1(x_k)) \in E'_i$ (as well as the edge on $\mathcal{R}_2$). If $x_k \notin \mathcal{C}'_i$, then $(x_k, \mathcal{R}_1(x_k)) \in E'_i$. (iii) If $x_k \notin S'$ and $x_k \in V_2$, then $x_{k+1} = \mathcal{R}_2(x_k)$ by construction of $E_{S'}$, and $x_k \notin \mathcal{C}'_i$. Thus, $(x_k, \mathcal{R}_2(x_k)) \in E'_i$. To summarize these three cases, all the arcs of the loop on $E_{S'}$ are also included in the arcs of E'_i, thus G'_i contains a loop. However, this is impossible by construction of $\mathcal{C}'_i$. Thus, by contradiction, there is no loop on $G_{S'}$.

- Suppose now that this loop spans several strongly connected components, including $\mathcal{C}_i$ and $\mathcal{C}_j$, with $i \neq j$. For all node x_m of the loop (with $(x_m, x_{m+1}) \in E_{S'}$), let us show that $(x_m, x_{m+1}) \in E'$. (i) If $x_m \in S'$, then $x_{m+1} = \mathcal{R}_2(x_m)$, and $x_m \in \mathcal{C}'_i$, which means that $x_m \in V_1$. Thus, $(x_m, \mathcal{R}_2(x_m)) \in E'$ by construction of E'. (ii) If $x_m \notin S'$ and $x_m \in V_1$, then $x_{m+1} = \mathcal{R}_1(x_m)$. Thus, $(x_m, x_{m+1}) \in E'$. (iii) If $x_m \notin S'$ and $x_m \in V_2$, then $x_{m+1} = \mathcal{R}_2(x_m)$. Thus, $(x_m, x_{m+1}) \in E'$. To summarize these three cases, for all x_m of the loop, $(x_m, x_{m+1}) \in E'$, so there exists a loop in G' between a node of $\mathcal{C}_i$ and a node of $\mathcal{C}_j$, which means that $\mathcal{C}_i$ and $\mathcal{C}_j$ are the same strongly connected component, which is impossible.

Let us show now that for any node x, d is reachable from x in $G_{S'}$. By contradiction, let us suppose that there is a node x from which d is not reachable. Let us consider the path starting from x in $G_{S'}$. Either there exists a node on the path that has no next-hop on $G_{S'}$, or the path has a loop. We just proved that there is no loop in $G_{S'}$, so there has to be a node y without a next-hop. By construction of $G_{S'}$, we can see that all nodes $y \in S' \cup (V_1 \backslash S') \cup V_2 = V$ have a next-hop, so d is reachable from any node x in $G_{S'}$. This completes the proof. □

Figure 3 shows an example on a graph of nine nodes, where destination is node f. The routing protocols $\mathcal{R}_1$ and $\mathcal{R}_2$ are shown on Figure 3(a). To build the first valid step, all routing arcs are considered. The strongly connected components of the resulting graph are the following: $\mathcal{C}_1 = \{f\}$, $\mathcal{C}_2 = \{b, c\}$ and $\mathcal{C}_3 = \{a, d, e, g, h, i\}$. The following sets verify the property of the theorem: $\mathcal{C}'_1 = \{f\}$, $\mathcal{C}'_2 = \{b\}$ and $\mathcal{C}'_3 = \{d, g, h, i\}$. Indeed, it can be verified that none of the graphs G'_i (for $i \in \{1, 2, 3\}$) has a loop (this can also be seen on the graph shown on Figure 3(b)). Thus, the step $S_1 = \{b, d, f, g, h, i\}$ is a valid first step. To build the second valid step, the strongly connected components of the graph shown on Figure 3(b) are computed. They are the following: $\mathcal{C}_1 = \{a\}$, $\mathcal{C}_2 = \{b\}$, ..., $\mathcal{C}_9 = \{i\}$. The following sets verify the property of the theorem: $\mathcal{C}'_1 = \{a\}$, $\mathcal{C}'_3 = \{c\}$, $\mathcal{C}'_5 = \{e\}$, and $\mathcal{C}'_i = \emptyset$ otherwise. The second step is $S_2 = \{a, c, e\}$, and the resulting graph is shown on Figure 3(c). After this step, all nodes route according to $\mathcal{R}_2$. Thus, $\mathbb{S} = (S_1, S_2)$ is a loop-free sequence.

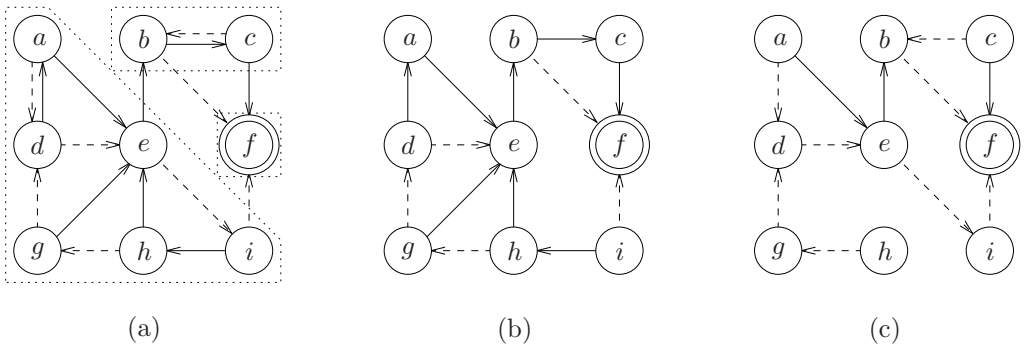


Figure 3: Sequence based on strongly connected components. (a) The graph has three strongly connected components $\mathcal{C}_1 = \{f\}$, $\mathcal{C}_2 = \{b, c\}$, $\mathcal{C}_3 = \{a, d, e, g, h, i\}$. (b) The first step is $S_1 = \{b, d, f, g, h, i\}$, and only the nodes of S_1 can route according to $\mathcal{R}_1$ or $\mathcal{R}_2$. (c) The second step is $S_2 = \{a, c, e\}$, and only the nodes of S_2 can route according to $\mathcal{R}_1$ or $\mathcal{R}_2$.

2.2. Building large steps is NP-hard

In the following, we show that building a sequence with a large first step is NP-hard, by reduction of the feedback vertex set problem.

Definition 3 (Feedback vertex set)**.** *The feedback vertex set (FVS) problem takes as input a graph $G = (V, E)$ and an integer k. It consists in determining whether there exists $X \subset V$ such that $|X| \leq k$ and $G - X$ is acyclic (that is, $G_X = (V_X, E_X)$ is loop-free, with $V_X = V \backslash X$ and $E_X = \{(x, y) \in E$ such that both $x \in V_X$ and $y \in V_X\}$). FVS is known to be NP-complete in both undirected and directed graphs [5].*

Definition 4 (Large valid first step)**.** *The large valid first step (LVFS) problem takes as input a graph $G = (V, E)$, a destination d, two routing protocols $\mathcal{R}_1$ and $\mathcal{R}_2$, and an integer k. It consists in determining whether there exists a valid step S with $V_1 = V$ and $V_2 = \emptyset$ such that $|S| \geq k$.*

Theorem 2. *LVFS is an NP-complete problem.*

Proof of Theorem 2. To prove Theorem 2, we use a polynomial reduction of FVS. The proof starts by showing that LVFS is in NP, shows how FVS can be reduced into LVFS, and shows that this reduction is polynomial.

Let us first show that LVFS is in NP, that is, let us show that a solution of LVFS can be verified in polynomial time. To check that a solution of LVFS is valid, we have to determine whether step S is valid, that is, whether there are routing loops when all nodes of $V \backslash S$ use $\mathcal{R}_1$ and nodes of S use $\mathcal{R}_1$ or $\mathcal{R}_2$. To verify this, we build the graph $G_s = (V, E_s)$, where E_s contains (i) for all $n \in V \backslash S$, $(n, \mathcal{R}_1(n))$, and (ii) for all $n \in S$, $(n, \mathcal{R}_1(n))$ and $(n, \mathcal{R}_2(n))$. If there is no loop in G_s, LVFS is valid. The loop verification can be performed by counting the number of strongly connected components of G_s using Tarjan's strongly connected components algorithm [6]: if the number of strongly connected components is different from $|V|$, there is a loop (or, alternatively, if there is a strongly connected component with two or more nodes). This can be done in $\mathcal{O}(|V| + |E_s|)$, which is polynomial.

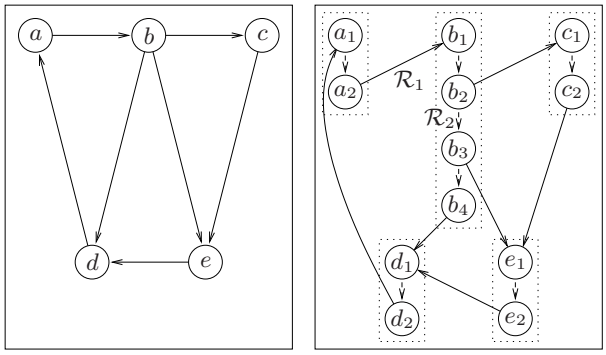
Let us now show how FVS can be reduced into LVFS.

- Let $(G = (V, E), k)$ be an instance of FVS, and let us build a corresponding instance $(G', z, \mathcal{R}_1, \mathcal{R}_2, |V| - k)$ for LVFS, where z is the destination of $\mathcal{R}_1$ and $\mathcal{R}_2$. Let us denote by $\delta(n)$ the outgoing degree of n in G. Each node n of G is transformed into $\delta(n) + 1$ nodes in G', denoted by n_i, with $i \in [1; \delta(n) + 1]$. For each node n of G, there is an arc (n_1, n_2) in G' labeled with $\mathcal{R}_2$. For each arc (n, m) in G, with m being the j-th neighbor of n in G, there is an arc (n_j, m_1) in G' labeled with $\mathcal{R}_1$. Then, an extra node z is added into G', and is considered the destination of $\mathcal{R}_1$ and $\mathcal{R}_2$ of LVFS. Finally, for every node n_i of G' that has only one arc labeled with $\mathcal{R}_1$, an arc (n_i, z) labeled with $\mathcal{R}_2$ is added, and for every node n_i of G' that has only one arc labeled with $\mathcal{R}_2$, an arc (n_i, z) labeled with $\mathcal{R}_1$ is added. Figure 4 shows the transformation of an instance of FVS into the corresponding instance of LVFS (without the node z for clarity). It can be noticed that both $\mathcal{R}_1$ and $\mathcal{R}_2$ in the new instance of LVFS do not cause loops and lead to z.

- Let us assume that FVS has a solution X of size k. We have to show that LVFS has a solution S of size $|V| - k$. Let us build step S in the following way: $S = V \backslash X$. By contradiction, let us suppose that step S is not valid, that is, there is a loop in $\mathcal{R}_{V \backslash S, S}$. By construction of G', the loop has to involve at least two nodes n_1 and m_1 of S. Thus, there is a path from n_1 to m_1 and from m_1 to n_1 in G' that involves nodes of S. In other words, there is a path from n to m and from m to n in G that does not involve nodes of X. Thus, there is a loop in $G - X$, which is impossible by construction of X. Thus, S is a valid step.

- Let us assume that FVS has no solution of size k. We have to show that LVFS has no solution of size $|V| - k$. Thus, for any step S (for LVFS) of size at most $|V| - k$, there exists a loop l that involves nodes of S. This loop also involves edges of $\mathcal{R}_2$, as $\mathcal{R}_1$ does not cause loops when considered alone. Let us build $S' \subset S$ such that S' contains all nodes of the loop l whose next-hop follows $\mathcal{R}_2$. This set S' violates Definition 1, thus S is not a valid step.

The reduction is polynomial. Indeed, the construction of the instance of LVFS from the instance of FVS is polynomial: if we write $G = (V, E)$ and $G' = (V', E')$, then $|V'| = 1 + |V| + |E|$ and $|E'| = 2|V'|$. □



(a) instance of FVS (b) instance of LVFS

Figure 4: Transformation of an instance of FVS into the corresponding instance of LVFS (destination node z is not shown).

2.3. Centralized heuristic for loop-free sequence

In this subsection, we describe our Centralized Heuristic for loop-free Sequence (CHS). We assume that a centralized entity knows the whole network topology, as well as $\mathcal{R}_1$ and $\mathcal{R}_2$ (that is, $\mathcal{R}_1(n)$ and $\mathcal{R}_2(n)$ is known, for each node n).

CHS is based on Algorithm 1. First, the set of strongly connected components of the graph is built. Then, each strongly connected component $\mathcal{C}_i$ is considered individually, and nodes are distributed into several steps according to Algorithm 2. We use a greedy approach: for each step S_j, we add nodes one by one into a set $\mathcal{C}'_{i,j}$, until it is not possible to add more nodes without violating the constraints of Theorem 1. Then, we move to the next step, until all nodes of $\mathcal{C}_i$ are in a step.

Algorithm 1 Main algorithm for CHS.

Require: $G = (V, E)$ a graph, $d \in V$ a destination, $\mathcal{R}_1$ and $\mathcal{R}_2$ two routing protocols for d
Ensure: $\mathbb{S}$ is a valid sequence
  $\mathcal{C} \leftarrow$ strongly connected components of G (with arcs resulting of $\mathcal{R}_1$ and $\mathcal{R}_2$)
  for $\mathcal{C}_i \in \mathcal{C}$ **do**
    if $|\mathcal{C}_i| = 1$ **then**
      the single node of $\mathcal{C}_i$ is in the first step S_1
    else
      $j \leftarrow 1$
      while there are nodes of $\mathcal{C}_i$ that are not in a step yet **do**
        find a suitable set $\mathcal{C}'_{i,j} \subset \mathcal{C}_i$ (see Algorithm 2)
        add nodes of $\mathcal{C}'_{i,j}$ to step S_j
        $j \leftarrow j + 1$
      end while
    end if
  end for
  return $\mathbb{S} = (S_1, \ldots, S_{j-1})$

The worst-case complexity of CHS is $\mathcal{O}(|V|^4)$. Indeed, the strongly connected components of a graph can be computed in $\mathcal{O}(|V| + |E|)$ with Tarjan's algorithm [6]. Note that in our graphs, $|E| \leq 2|V|$ as each node has at most two outgoing arcs (one with $\mathcal{R}_1$ and one with $\mathcal{R}_2$). For each strongly connected component $\mathcal{C}_i$,

Algorithm 2 Computation of one step for $\mathcal{C}_i$ in CHS

Require: $\mathcal{C}_i$ a strongly connected component (with at least two nodes), list of previous steps
Ensure: $\mathcal{C}'_{i,j}$ satisfies the property of Theorem 1

```
C'_{i,j} ← ∅
end ← false
while not end do
  end ← true
  for node n in C_i do
    if n is not already in a step and n ∉ C'_{i,j} then
      build G' with the nodes of G and no edges
      for each node m that has been added in a previous step, add arc (m, R_2(m)) to G'
      for each node m ∈ C'_{i,j} ∪ {n}, add arcs (m, R_1(m)) and (m, R_2(m)) to G'
      for each other node m ≠ d, add arc (m, R_1(m)) to G'
      if G' does not contain a loop then
        C'_{i,j} ← C'_{i,j} ∪ {n}
        end ← false
      end if
    end if
  end for
  return C'_{i,j}
end while
```

there are at most $|\mathcal{C}_i|$ resulting steps (as there is at least one node per step). Computing the set $\mathcal{C}'_{i,j}$ requires examining at most $|\mathcal{C}_i|^2$ combinations of nodes, and each examination requires building G' and determining if it has a loop, which can be done in $\mathcal{O}(|\mathcal{C}_i|)$. As the set of strongly connected components partitions the graph, we obtain the overall complexity of $\mathcal{O}(|V|^4)$.

2.4. Distributed protocol for loop-free sequence

In this subsection, we describe our Distributed Protocol for loop-free Sequence (DPS). We assume that each routing protocol has a protocol ID. This ID is used to differentiate protocols $\mathcal{R}_1$ and $\mathcal{R}_2$. DPS works as follows:

- Initially, all nodes route according to $\mathcal{R}_1$.
- A control message is sent to each node (in arbitrary order and at arbitrary time); upon receiving this message, each node configures the new routing protocol $\mathcal{R}_2$, but continues to forward packets according to $\mathcal{R}_1$. The routing protocol $\mathcal{R}_2$ is said to be pending for this node.
- Destination d is allowed to perform the transition to $\mathcal{R}_2$ as soon as $\mathcal{R}_2$ is pending.
- Regularly, a node $n \neq d$ that has a pending routing protocol sends a control message to its neighbor $\mathcal{R}_2(n)$ to request for its protocol ID. The neighbor replies with a control message indicating whether it uses $\mathcal{R}_1$ or $\mathcal{R}_2$ (by including in the reply the protocol ID).
- When a node $n \neq d$ receives from its neighbor $\mathcal{R}_2(n)$ a reply containing the protocol ID of $\mathcal{R}_2$, n is allowed to perform the transition to $\mathcal{R}_2$. To do this, n starts forwarding packets according to $\mathcal{R}_2$, and can remove $\mathcal{R}_1$ from memory.

Proposition 1. *DPS achieves a loop-free sequence.*

Proof. Let us denote by $\mathbb{S}$ the order in which nodes perform the transition from $\mathcal{R}_1$ to $\mathcal{R}_2$ in DPS. We assume here that each step of $\mathbb{S}$ contains a single node, as it is not possible for two different events to occur at the same time. In the following, we show that $\mathbb{S}$ contains all the nodes of V and that $\mathbb{S}$ is a loop-free sequence.

Let us show that $\mathbb{S}$ contains all nodes of V. Since all nodes will eventually receive the configuration message, we have to show that it is not possible for a node to have $\mathcal{R}_2$ pending indefinitely. By contradiction, let us assume that n has $\mathcal{R}_2$ pending indefinitely, and let us build the path $p = (n_0, n_1, n_2, \ldots)$ where $n_0 = n$ and for all i, $n_{i+1} = \mathcal{R}_2(n_i)$. If there exists i such that n_i has $\mathcal{R}_2$ pending indefinitely, then n_{i+1} also has $\mathcal{R}_2$ pending indefinitely (according to the protocol description). Thus, by induction on i, all nodes of p have $\mathcal{R}_2$ pending indefinitely. Since p eventually reaches d ($\mathcal{R}_2$ is a loop-free routing protocol to d), d has to be pending indefinitely, which is not possible. Thus, we have a contradiction, and we can conclude that $\mathbb{S}$ contains all nodes of V.

Let us show that $\mathbb{S} = (S_1, S_2, \ldots, S_{|V|})$ is a loop-free sequence. We have to show that for all $i \in [1; |V|]$, S_i is a loop-free step on partition (V_1^i, V_2^i), with $V_2^i = S_1 \cup \ldots \cup S_{i-1}$ and $V_1^i = V \backslash V_2^i$. Let us show this by induction on i.

- Let us show that S_1 is a loop-free step. Since d is the first node that can perform the transition to $\mathcal{R}_2$, $S_1 = \{d\}$. Moreover, since d is the destination, $\mathcal{R}_{V,\emptyset} = \mathcal{R}_{V\backslash\{d\},\{d\}}$. We know that $\mathcal{R}_{V,\emptyset} = \mathcal{R}_1$ is loop-free. Then, for all $S' \subset \{d\}$, $\mathcal{R}_{V\backslash S',\emptyset\cup S'}$ is loop-free.

- Let us assume that S_i is a loop-free step, and let us show that S_{i+1} is a loop-free step. Let us denote by n_{i+1} the node such that $S_{i+1} = S_i \cup \{n_{i+1}\}$. We have to show that for all $S' \subset \{n_{i+1}\}$, $\mathcal{R}_{V_1^i\backslash S', V_2^i \cup S'}$ is loop-free. This is true for $S' = \emptyset$, since S_i is a valid step. Let us now consider the case where $S' = \{n_{i+1}\}$, and let us show that $\mathcal{R}_{V_1^i \cup \{n_{i+1}\}, V_2^i \backslash \{n_{i+1}\}}$ is loop-free. Let n be any node of the network, and let us build the path $p = (n_0, n_1, n_2, \ldots)$ with $n_0 = n$, $n_{j+1} = \mathcal{R}_1(n_j)$ if $n_j \in V_1^{i+1}$ and $n_{j+1} = \mathcal{R}_2(n_j)$ otherwise. By contradiction, let us assume that p has loops. Then, p has to go through n_{i+1} since S_i is loop-free. By construction of $\mathbb{S}$, the path from n_{i+1} follows only nodes of $\mathcal{R}_2$ (since a node performs the transition to $\mathcal{R}_2$ only after its next-hop on $\mathcal{R}_2$ has performed the transition to $\mathcal{R}_2$ too). Thus, the path from p reaches d as $\mathcal{R}_2$ is loop-free. The path from n to p follows only nodes of $\mathcal{R}_1$, and cannot have loop either. Thus, p is loop-free. By contradiction, we have shown that S_{i+1} is a loop-free step.

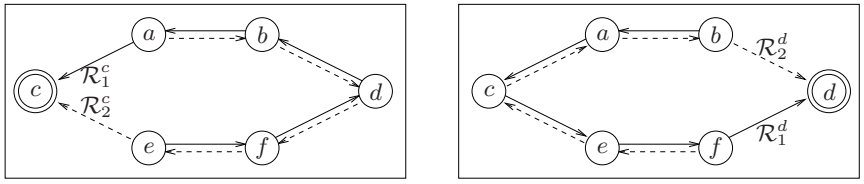
□

In DPS, nodes perform the transition independently, based only on the routing protocol used by the next-hop on $\mathcal{R}_2$. If the decision to perform the transition takes the same time for all nodes, we can also consider that nodes perform the transition according to their distance to the destination: nodes at one hop of the destination can perform the transition together, then nodes at two hops of the destination can perform the transition together, etc. In the following, we consider that nodes of DPS perform the transition by steps, where each step contains all the nodes at the same distance to the destination. We also consider that the destination is in the same step as the nodes that are one hop away from the destination, as the destination cannot cause loops while performing the transition.

2.5. Changing routing protocols for several destinations

When there are several destinations for the routing protocols, it might not be possible to change the routing protocol for all destinations simultaneously. Figure 5 provides such an example on a topology of six nodes with two destinations c and d. Figure 5(a) shows the routing protocols $\mathcal{R}_1^c$ and $\mathcal{R}_2^c$ for destination c, and Figure 5(b) shows the routing protocols $\mathcal{R}_1^d$ and $\mathcal{R}_2^d$ for destination d. If node a switches to $\mathcal{R}_2^c$ and to $\mathcal{R}_2^d$ simultaneously, a loop occurs between a and b for packets going to either c and d. If node b switches to $\mathcal{R}_2$ for both destinations, a loop between b and d occurs for packets going to c, and a loop between a and b occurs for packets going to d. If node c switches to $\mathcal{R}_2$ for both destinations, a loop between a and c occurs for packets going to d. Similarly, a loop occurs if nodes d, e or f perform the transition of their routing protocol to $\mathcal{R}_2$.

To handle several destinations, it is possible to apply our mechanisms for each destination independently, until nodes have performed the transition of their routing protocol for every destination. We leave this as future work, and focus in this paper on a single destination.



(a) (b)

Figure 5: With these two routing protocols $\mathcal{R}_1$ and $\mathcal{R}_2$ for the two destinations c (shown on the left) and d (shown on the right), it is not possible to change the routing protocol for the two destinations simultaneously without causing loops.

3. Performance evaluation

In this section, we compare CHS and DPS through simulations and experiments. First, we quantify the probability of loop occurrences when using arbitrary routing protocols, with arbitrarily transitions (that is, without any mechanism to avoid transient loops). Then, we compute by simulation the number of steps required to perform the transition, for our CHS and DPS. Finally, we evaluate the time required by our implementation of CHS and DPS on motes to perform the transition.

3.1. Simulation on loop occurrences with arbitrary transitions

We consider that a loop occurs on a given topology if a packet can enter a routing loop when nodes on the path decide arbitrarily to route according to $\mathcal{R}_1$ or $\mathcal{R}_2$. For instance, we consider that there is a loop in the topology shown on Figure 1, because b might route according to $\mathcal{R}_1$ while c routes according to $\mathcal{R}_2$. Notice that even if there is a loop occurrence in a topology, some nodes might be able to send packets to the destination without loops. Thus, our metric refers to the risk of a possible loop.

We generated random graphs, and we used routing protocols based on shortest paths (for various weights on links, as described later). Each graph is generated by deploying nodes uniformly at random in an area of 100m×100m, and by ensuring that the deployment is connected (with a given communication range). All results are averaged over 1000 simulations.

Figure 6 shows the percentage of loop occurrences as a function of the node number, for two communication ranges (20m and 30m). The routing protocol $\mathcal{R}_1$ is a shortest path protocol based on the number of hops to reach the destination. The routing protocol $\mathcal{R}_2$ is a shortest path protocol based on a random weight on each link, chosen uniformly in $[1; 100]$. We use this random weight to model link metrics such as loss. The percentage of loop occurrences increases with the number of nodes. Even when the number of nodes is small (for instance, 50), the percentage of loop occurrences is above 50%, which means that loops are likely to occur. This comes from the fact that the two protocols build uncorrelated paths due to the random weight of $\mathcal{R}_2$.

Figure 7 shows the percentage of loop occurrences as a function of the node number, for two communication ranges. Both routing protocols $\mathcal{R}_1$ and $\mathcal{R}_2$ are shortest path protocols based on a random weight on each link, chosen uniformly in $[1; 100]$. We use this random weight to model two independent metrics, such as loss for $\mathcal{R}_1$ and delay for $\mathcal{R}_2$, when nodes have independent wake-up schedules. The percentage of loop occurrences is about 100% for all topologies. This is due to the fact that $\mathcal{R}_1$ and $\mathcal{R}_2$ yield to different routing decisions, thus paths have a very low correlation.

Figure 8 shows the percentage of loop occurrences as a function of the node number, for two communication ranges. The routing protocol $\mathcal{R}_1$ is based on a random weight on each link, chosen uniformly in $[1; 100]$. The routing protocol $\mathcal{R}_2$ uses a correlated weight: if w is the weight of the link in $\mathcal{R}_1$, the weight of the same link in $\mathcal{R}_2$ is chosen uniformly in $[w - 5; w + 5]$. Thus, $\mathcal{R}_1$ and $\mathcal{R}_2$ are highly correlated. This reduces the overall percentage of loops, for both communication ranges, especially for small number of nodes. When the communication range is larger or when the number of nodes is larger, the network density increases. When the density is high, the two routing protocols have many choices to compute the paths, and the resulting shortest paths exhibit low correlations (even if the weights are correlated).

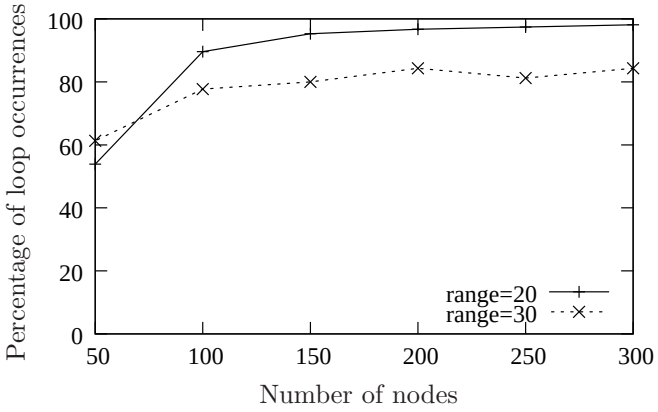


Figure 6: When $\mathcal{R}_1$ is based on a hop-count metric and $\mathcal{R}_2$ is based on a random metric, the percentage of loop occurrences is high, due to the low correlation of the metrics of $\mathcal{R}_1$ and $\mathcal{R}_2$.

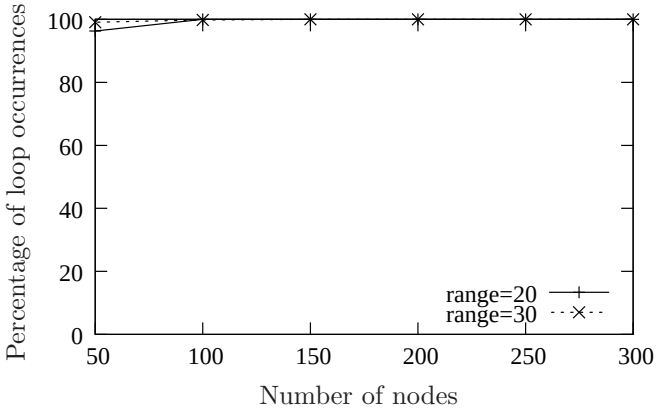


Figure 7: When $\mathcal{R}_1$ and $\mathcal{R}_2$ are both based on a random metric, the percentage of loop occurrences is very high, again due to the very low correlation of the metrics of $\mathcal{R}_1$ and $\mathcal{R}_2$.

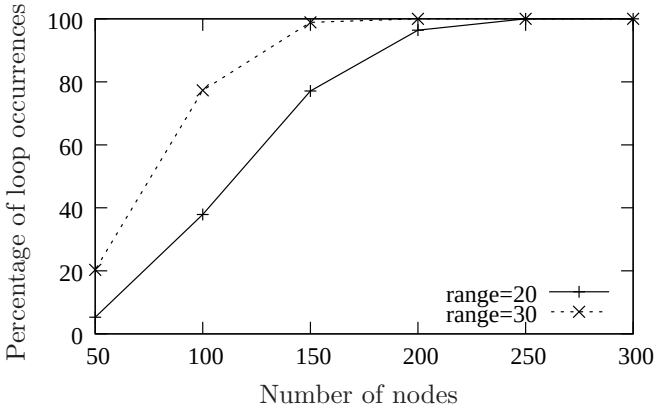


Figure 8: When $\mathcal{R}_1$ is based on a random metric and $\mathcal{R}_2$ is based on a 5%-deviation of the metric of $\mathcal{R}_1$, the percentage of loop occurrences is high for topologies with a large number of nodes, despite the high correlation of the metrics.

3.2. Simulations on number of steps with CHS and DPS

We consider now the case where $\mathcal{R}_2$ is based on a 5%-deviation of $\mathcal{R}_1$ (as shown on Figure 8), which is the most realistic case in our opinion.

We compute the number of steps for CHS and DPS for 1000 repetitions. In repetitions without loop occurrences, CHS is able to perform the transition in a single step. However, DPS still requires several steps, as the number of steps of DPS depends on the maximum distance of nodes to the destination.

Figure 9 shows the average number of steps for both protocols, for a communication range of 20m (on the left) and of 30m (on the right). The figure also depicts the minimum and maximum number of steps obtained during the 1000 repetitions. The average number of steps for CHS increases slightly with the number of nodes, and is about 2 steps for all repetitions, which indicates that CHS is able to solve most loops efficiently, with only a single extra step. The average number of steps for DPS is larger than with CHS because the number of steps with DPS is the maximum depth of the network. However, our simulation results show that the average number of steps with DPS is only about three times larger than with CHS, although DPS is fully distributed.

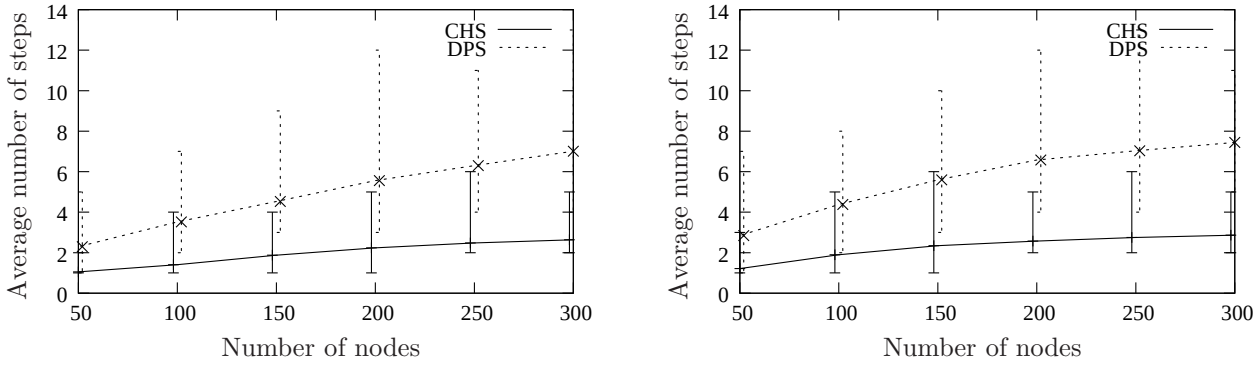


Figure 9: CHS is able to perform the transition in about two steps on average, when the communication range is 20m (on the left) or 30m (on the right). DPS requires about three times more steps to perform the transition.

3.3. Experiments on transition duration with CHS and DPS

In this subsection, we evaluate our protocols using a real testbed of TelosB motes. We implemented the centralized heuristic, CHS, and the distributed protocol, DPS, using NesC/TinyOS. In the following, we first describe the implementation of these two protocols. Then, we describe the experimental results.

3.3.1. Details on protocol implementation

CHS is implemented as follows. The destination knows in advance the network topology, and is able to compute all the steps before starting the transition process. The destination initiates the transition process by sending a `SWITCH` message with the list of nodes that have to perform the transition at the first step. Each node n receiving this message broadcasts it, and performs the transition immediately to $\mathcal{R}_2$ if n is in the list. When a node has perform the transition to $\mathcal{R}_2$, it sends a `CONFIRM` message to the destination in unicast, according to $\mathcal{R}_2$. To avoid collisions, both `SWITCH` and `CONFIRM` messages are sent with a random delay. When the destination receives all the expected confirmations for a given step, it starts the next step by sending a new `SWITCH` message with the new step number and the corresponding list of nodes. If the destination has not received all the confirmations after a given delay, it rebroadcasts the `SWITCH` message with the address of the remaining nodes of the current step.

DPS is implemented as follows. Each node broadcasts a `HELLO` message to its neighbors within a random period. This message contains the number of the routing protocol used (that is, 1 for $\mathcal{R}_1$ and 2 for $\mathcal{R}_2$). The destination initiates the transition process by changing to $\mathcal{R}_2$ silently. When a node n receives a `HELLO` message from its neighbor $\mathcal{R}_2(n)$, n performs the transition to $\mathcal{R}_2$ only if $\mathcal{R}_2(n)$ has already performed the transition to $\mathcal{R}_2$.

3.3.2. Experimental results

We used five topologies of ten nodes, depicted on Figure 10. For all these topologies, the next-hop according to $\mathcal{R}_1$ (represented with solid lines) and $\mathcal{R}_2$ (represented with dashed lines) has been statically defined in each node. Recall that the destination is denoted by a double circle. Figure 10(a) shows a linear topology. It corresponds to a worst-case transition: both protocols generate the following sequence of nine steps $\mathbb{S} = (\{a, f\}, \{g\}, \{h\}, \{i\}, \{j\}, \{e\}, \{d\}, \{c\}, \{b\})$. Figure 10(b) shows a tree topology. Both protocols generate the following sequence of two steps: $\mathbb{S} = (\{a, d, g, h, i, j\}, \{b, c, e, f\})$. Figure 10(c) shows a grid topology. It corresponds to a best-case transition for CHS, as all nodes can perform the transition in one step, without causing loops. DPS generates the following sequence of five steps: $\mathbb{S} = (\{i, j\}, \{f, h\}, \{c, e, g\}, \{b, d\}, \{a\})$. Figure 10(d) shows a mesh topology with two arbitrary routing protocols. CHS generates the following sequence of four steps: $\mathbb{S} = (\{a, b, e, g, h, j\}, \{d, i\}, \{f\}, \{c\})$. DPS generates the following sequence of six steps: $\mathbb{S} = (\{g, j\}, \{i\}, \{e, f\}, \{b, c, h\}, \{a\}, \{d\})$. Figure 10(e) shows another mesh topology with two arbitrary routing protocols. CHS generates the following sequence of three steps: $\mathbb{S} = (\{a, g, i, j\}, \{b, d, f\}, \{c, e, h\})$. DPS generates the following sequence of eight steps: $\mathbb{S} = (\{g, j\}, \{d\}, \{h\}, \{i\}, \{f\}, \{c, e\}, \{a\}, \{b\})$.

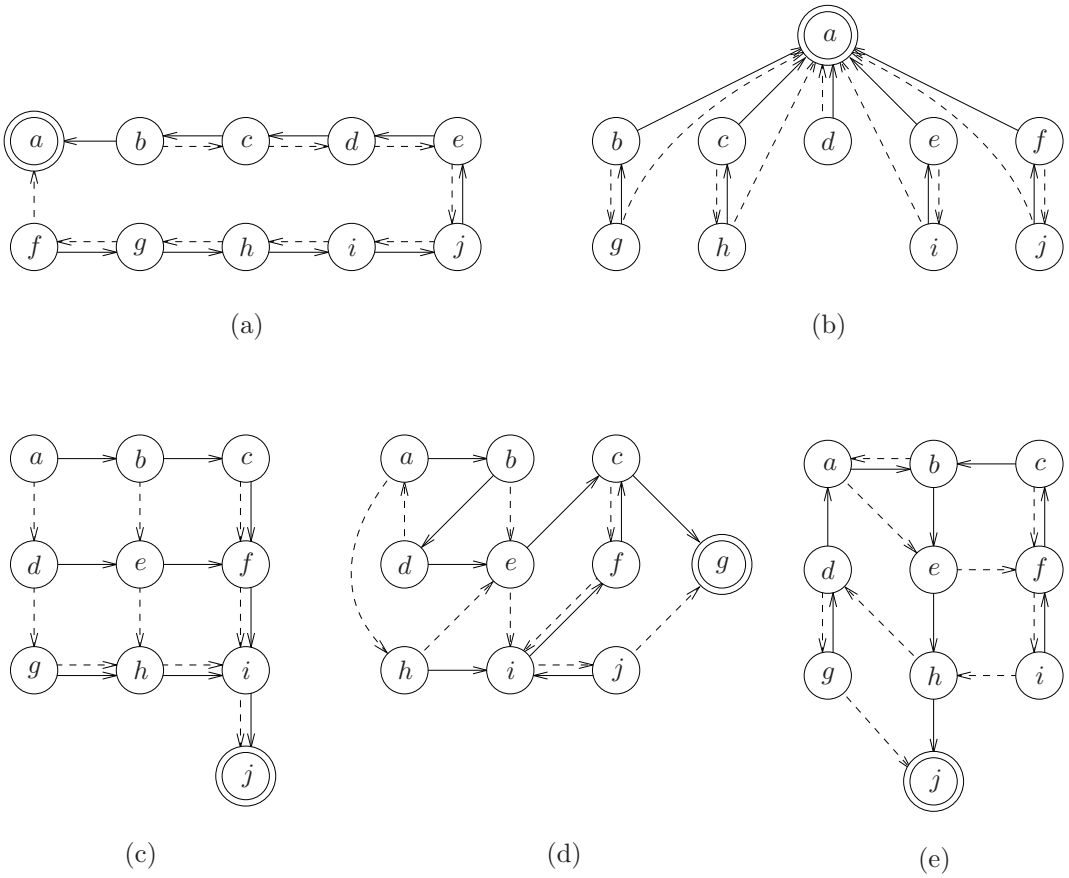


Figure 10: Five topologies for the experimental testbed.

We used the following protocol settings. For CHS, the random delay (in seconds) for both `SWITCH` and `CONFIRM` messages is chosen within a period of $[0.5; 1]$. The delay before the retransmission of a `SWITCH` message (when some `CONFIRM` messages are lost) is set to 8 s. For DPS, `HELLO` messages are sent randomly within a period of $[5.5; 6.5]$ (in seconds). These settings were chosen so that both protocols take a similar time for the transition on the topologies illustrated on Figure 10(a) and Figure 10(b), as both protocols generate the same steps for these two topologies.

Table 1 shows the experimental results, averaged over 10 repetitions. The granularity of the results

is 1 s, and the standard deviation of experimental transition duration is also shown. For the topology of Figure 10(a), nine steps are required for both protocols. The two protocols perform the transition in about 26 s. Indeed, the expected duration in DPS for a step (between the changing of a node and the sending of the `HELLO`) is half the average `HELLO` period, that is 3 s. As there are nine steps, the expected transition duration is about 27 s. For the topology of Figure 10(b), two steps are required for both protocols. They perform the transition in about 8 s, and the expected transition duration is about 6 s. For the topology of Figure 10(c), CHS achieves the transition in only 2 s, as all nodes can perform the transition during the first step. Note that for this topology, the transition duration does not take into account the possible loss of `CONFIRM` messages, as the destination does not need to wait for the confirmations to start another step. DPS shows a larger transition duration due to a larger number of steps. For the topology of Figure 10(d), CHS achieves a delay of about 28 s. We noticed in a few repetitions that some `CONFIRM` messages were lost by the destination during the first step. This increases latency because the destination has to retransmit a `SWITCH` message with the same step number. This is probably caused by the large number of nodes in the first step, generating collisions of the `CONFIRM` messages. This phenomenon impacts the standard deviation of the results. For the topology of Figure 10(e), both protocols have the same average transition duration. The relatively large transition duration for CHS is due to the relatively large number of nodes per step (recall that the `CONFIRM` messages are sent in unicast according to $\mathcal{R}_2$, and thus have to be retransmitted several times).

Topology		Fig. 10(a)	Fig. 10(b)	Fig. 10(c)	Fig. 10(d)	Fig. 10(e)
CHS	Average time	25.0 s	9.2 s	2.0 s	28.0 s	15.9 s
	Standard dev.	0 s	0.42 s	0 s	4.18 s	0.31 s
	Number of steps	9	2	1	4	3
DPS	Average time	26.2 s	7.6 s	12.8 s	22.2 s	16.0 s
	Standard dev.	4.89 s	0.96 s	2.04 s	3.08 s	3.94 s
	Number of steps	9	2	5	6	8

Table 1: Experimental transition durations on motes.

4. Related work

In this section, we first describe architectures where routing loops might occur. Then, we present some solutions from the literature to avoid routing loops occurring during the transition from one protocol to another.

4.1. Networks with risk of loop occurrences

Several routing protocols that combine different routing decisions have been proposed in the literature. In [7], the authors propose to combine a reactive routing protocol with a greedy geographical routing protocol. When a packet has to be forwarded, the reactive protocol establishes the whole route to the destination. The geographical protocol is used when the next-hop according to the reactive protocol becomes unreachable. Routing loops can occur if the geographical protocol forwards packets to a node that uses the reactive protocol. In [8], a routing protocol R_d that reduces delay is combined with a routing protocol R_e that reduces energy. R_d and R_e are used depending on the traffic produced by the application: urgent packets are forwarded according to R_d, while periodic packets are forwarded according to R_e. Routing loops can occur if an urgent packet reaches a node that has a limited energy and uses R_e. In [9], packets are given a priority based on traffic type. The next-hop of a packet is computed according to several parameters, including packet priority, number of hops to the destination, link quality for the next-hop, residual energy for the next-hop, load of the next-hop, etc. Routing loops can occur if the parameters used by a node n_1 are different from the parameters used by another node n_2 on the path to the destination.

Multi-purpose WSNs [10] have been proposed to consider a single WSN deployment to support several applications. The main advantage of multi-purpose WSNs is that the cost of deployment is shared by all the

applications. Several researchers have proposed protocols for multi-purpose WSNs [11, 12, 13, 14]. In such networks, several routing protocols are used simultaneously, because the large amount of applications yield to different requirements that cannot be met by a single routing protocol. However, dealing with several routing protocols might cause routing loops when the choice of the routing protocol is made locally by each node.

4.2. Avoiding routing loops

The problem of avoiding transient loop in a network has been rarely studied in the literature. We summarize here the main works.

In [15, 16], nodes use two routing protocols $\mathcal{R}_1$ and $\mathcal{R}_2$ alternatively: a node forwards packets according to a repetitive cycle composed of two periods p_1 and p_2. During p_1, the node forwards packets according to $\mathcal{R}_1$, and during p_2, the node forwards packets according to $\mathcal{R}_2$. Routing loops can occur if the decision to forward a packet according to $\mathcal{R}_1$ or $\mathcal{R}_2$ is made locally by each node. In [15], properties of pairs of routing protocols are studied, and three categories are identified: (i) compatible routing protocols, which do not yield to routing loops, (ii) delayable routing protocols, where nodes might avoid loops based on the knowledge of the distance functions of the two protocols, and (iii) combined routing protocols, when the distance functions of the two protocols are not known or hard to compute locally by the nodes. In [16], two heuristics were proposed to avoid loops or reduce their occurrences. In the first heuristic, loops are avoided by forbidding some nodes to forward packets during one period p_i. In the second heuristic, a probabilistic approach is used to reduce the risk of loops: nodes that could lead to potential loops choose randomly whether to forward or to hold packets. The difference between [15, 16] and this paper is that we consider here that the transition between $\mathcal{R}_1$ and $\mathcal{R}_2$ is final: once a node has performed the transition to $\mathcal{R}_2$, it does not change back to $\mathcal{R}_1$. Thus, only the compatible property of [15] is applicable, and the two heuristics of [16] cannot be applied in this context as they rely on the alternation of $\mathcal{R}_1$ and $\mathcal{R}_2$.

In [3], the authors show that most routing protocols can produce transient routing loops after a topological change. They show that such loops can be avoided by having routers process routing updates according to a specific order. Their mechanism is able to deal with link failures, new links, or updates on link metrics. The differences with this paper are the following: (i) we consider arbitrary protocols for $\mathcal{R}_1$ and $\mathcal{R}_2$, while [3] considers a single routing protocol on two similar topologies, and (ii) we reduce the number of steps required for the transition, while [3] provides an ordering of updates that does not cause loops.

In [2, 4], the authors show that ordering the routing updates yields to additional message overhead and increases the convergence delay. They avoid transient routing loops by exploiting the existence of one forwarding table per interface. Messages arriving through unexpected interfaces are discarded, because they indicate a discrepancy between the view of the router and its neighbors. Once all routers have the same view of the topology, the protocol can converge and produce loop-free routes. The main difference with this paper is that we do not drop packets explicitly to avoid loops.

In [17, 18], the authors show that transient routing loops that occur after topological changes can be avoided by applying a sequence of topology updates. Between two topology updates, the same routing protocol is used, but some link values are modified, which result into changes in routing decisions. They propose a protocol that minimizes the number of topology updates. The difference with this paper are the following: (i) we consider two different routing protocols on a single topology, while [17, 18] consider a single routing protocol on a sequence of updated topologies, (ii) we consider arbitrary routing protocols, while [17] considers changes on only one link and [18] considers changes concerning the links of a single router, and (iii) we reduce the number of steps to perform the transition, where each step is a set of nodes and each node appears only once overall, while [17, 18] minimizes the number of topology updates.

5. Conclusion

When a running routing protocol has to be changed, transient routing loops might occur in the network. In this paper, we quantified the percentage of loop occurrence on several scenarios. We showed that loops can be completely avoided by having nodes change from one protocol to another according to a sequence,

which depends on the topology and on the routing protocols. The sequence is a list of steps, where each step is a set of nodes that can perform the transition in arbitrary order. We proposed a centralized heuristic that builds such a sequence with a limited number of steps, and a distributed protocol that builds sequences with a larger number of steps, but without requiring the knowledge of the whole network topology. Simulation results show that the distributed protocol computes sequences that have about three times more steps than the centralized heuristic. However, experimental results show that the actual time required by the distributed protocol is often close to the time required by the centralized heuristic.

References

[1] P. Trakadas, T. Zahariadis, S. Voliotis, P. Karkazis, T. Velivassaki, L. Sarakis, Routing metric selection and design for multi-purpose WSN, in: Systems, Signals and Image Processing (IWSSIP), 2014 International Conference on, 2014, pp. 199–202.

[2] Z. Zhong, R. Keralapura, S. Nelakuditi, Y. Yu, J. Wang, C. N. Chuah, S. Lee, Avoiding transient loops through interface-specific forwarding, in: IWQoS, Vol. 3552 of Lecture Notes in Computer Science, Springer, 2005, pp. 219–232.

[3] P. Francois, O. Bonaventure, Avoiding transient loops during the convergence of link-state routing protocols, IEEE/ACM Transactions on Networking 15 (6) (2007) 1280–1292. doi:10.1109/TNET.2007.902686.

[4] S. Nelakuditi, Z. Zhong, J. Wang, R. Keralapura, C. N. Chuah, Mitigating transient loops through interface-specific forwarding, Computer Networks 52 (3) (2008) 593–609.

[5] M. R. Garey, D. S. Johnson, Computers and intracability. A guide to the theory of NP-completeness, 1979.

[6] R. E. Tarjan, Depth-first search and linear graph algorithms, SIAM Journal on Computing 1 (2) (1972) 146–160.

[7] R. Shirani, M. St-Hilaire, T. Kunz, Y. Zhou, J. Li, L. Lamont, Combined reactive-geographic routing for unmanned aeronautical adhoc networks, in: IWCMC (International Wireless Communications and Mobile Computing Conference), 2012.

[8] S. Moad, M. T. Hansen, R. Jurdak, B. Kusy, N. Bouabdallah, A. Ksentini, On balancing between minimum energy and minimum delay with radio diversity for wireless sensor networks, in: IFIP Wireless Days, 2012.

[9] M. Fonoage, M. Cardei, A. Ambrose, A QoS based routing protocol for wireless sensor networks, in: IPCCC (IEEE Performance Computing and Communications Conference), 2010.

[10] J. Steffan, L. Fiege, M. Cilia, A. P. Buchmann, Towards multi-purpose wireless sensor networks, in: Systems Communications, 2005, pp. 336–341.

[11] J. Steffan, M. Voss, Security mechanisms for multi-purpose sensor networks, in: GI Jahrestagung, Vol. 2, 2005, pp. 158–160.

[12] D. Manjunath, S. V. Gopaliah, A multi-purpose wireless sensor network for residential layouts, in: BWCCA (International Conference on Broadband and Wireless Computing, Communication and Applications), 2007, pp. 49–58.

[13] P. Javier del Cid, Optimizing resource use in multi-purpose WSNs, in: PerCom Workshops (Pervasive Computing Workshops), 2011, pp. 395–396.

[14] L. Sarakis, T. Zahariadis, H.-C. Leligou, M. Dohler, A framework for service provisioning in virtual sensor networks, EURASIP Journal on Wireless Communications and Networking (1) (2012) 135.

[15] N. El Rachkidy, A. Guitton, M. Misson, Avoiding routing loops in a multi-stack WSN, JCM (Journal of Communications) 8 (3) (2013) 751–761.

[16] N. El Rachkidy, A. Guitton, M. Misson, Improving routing performance when several routing protocols are used sequentially in a wsn, in: ICC (IEEE International Conference on Communications), 2013, pp. 1408–1413.

[17] F. Clad, P. Merindol, J.-J. Pansiot, P. Francois, O. Bonaventure, Graceful convergence in link-state IP networks: A lightweight algorithm ensuring minimal operational impact, IEEE/ACM Transactions on Networking 22 (1) (2013) 300–312. doi:10.1109/TNET.2013.2255891.

[18] F. Clad, P. Merindol, S. Vissicchio, J.-J. Pansiot, P. François, Graceful router updates for link-state protocols, in: IEEE ICNP, 2013.